MASFACT: CONTINUAL MULTI-AGENT TOPOLOGY LEARNING VIA GEOMETRY-AWARE POSTERIOR TRANSFER

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems (MAS) powered by large language models (LLMs) have emerged as a powerful paradigm for complex problem solving, where performance critically depends on the underlying inter-agent communication topology. However, existing topology generation methods mainly optimize for isolated tasks, while real-world deployments involve streams of evolving tasks, requiring previously effective collaboration patterns to be retained and reused rather than rediscovered or overwritten. We identify a previously underexplored failure mode, *topology forgetting*, in which adapting to new tasks shifts the topology generator away from communication structures required by earlier tasks. This issue stems from cross-task misalignment in both agent-level functional semantics and relational communication structures. To address this challenge, we propose **MASFACT**, a geometry-aware posterior transfer framework that preserves and reuses historical collaboration knowledge as transferable topology priors. We transfer these priors across task-specific agent spaces through Fused Gromov-Wasserstein optimal transport and perform PAC-Bayes-guided conservative posterior adaptation to balance task-specific plasticity with structural stability. Experiments across class-, domain-, and task-level continual settings demonstrate that MASFACT consistently improves average accuracy while reducing topology forgetting compared to strong topology generation and replay-based baselines, and can be seamlessly integrated with different MAS topology generators.

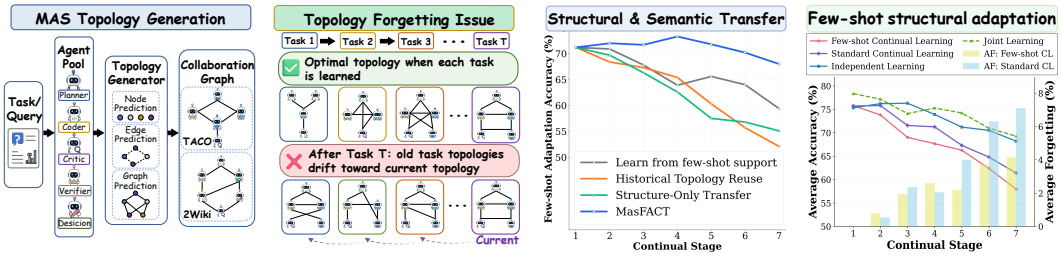
1 INTRODUCTION

Large language model (LLM)-based multi-agent systems (MAS) have emerged as a powerful paradigm for complex reasoning, planning, and decision-making tasks (Guo et al., 2024; Wu et al., 2023). A central factor governing their effectiveness is the underlying communication topology, which determines how agents coordinate, exchange information, and aggregate intermediate reasoning (Du et al., 2023; Shen et al., 2025; Hong et al., 2024). An effective topology can substantially improve reasoning quality and coordination efficiency while mitigating structural inefficiencies such as redundant communication and context dilution (Fan et al., 2026). Consequently, topology design has become a fundamental bottleneck in scaling reliable and adaptive LLM-based MAS.

Recent studies have explored automated MAS topology generation through search-based optimization and generative graph modeling (Leong et al., 2025), but largely assumed static task distributions or isolated-task optimization (Chen et al., 2024). In realistic deployments, MAS must continually adapt to evolving task streams, where previously effective collaboration structures should be retained and reused rather than repeatedly rediscovered or overwritten. As illustrated in Figure 1b, we call this failure mode *topology forgetting*: the topology generator gradually shifts away from the communication structures required by previous tasks when it adapts to new tasks, losing the ability to reconstruct them. Unlike standard parameter forgetting, topology forgetting stems from cross-task misalignment in topology space, where both relational structures and agent semantics may shift.

Therefore, we study continual MAS topology learning, where a topology generator must adapt to new tasks while preserving reusable collaboration topologies from prior tasks. Historical communication structures therefore should be treated not as isolated optimization outcomes, but as reusable structural

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Task Illustration (b) Topology Forgetting (c) Challenge 1 (d) Challenge 2

Figure 1: Overview of MAS topology and related challenges. (a) An illustration of MAS topology generation. (b) Optimization of current task leads to topology interference on previous tasks. (c-d) Two challenges of continual MAS topology learning task.

knowledge transferable across tasks. Existing topology optimization frameworks are not designed to address this setting: search-based methods overwrite past structures during optimization (Li et al., 2025; Wang et al., 2025), while generative topology models suffer from distribution drift without explicit structural memory (Leong et al., 2025; Fan et al., 2026). As a result, current approaches typically trade the short-term adaptation at the cost of degenerating long-horizon knowledge. This problem thus introduces two key challenges:

Challenge I: Joint structural-semantic transfer. MAS topologies encode both communication links and agent-level semantics (e.g. , assigned roles, functional capabilities and activation conditions), requiring alignment across both relational structure and functional semantics. As shown in Figure 1c, transferring structure alone yields suboptimal performance.

Challenge II: Few-shot adaptation under structural interference. Sparse feedback signals make naïve updates prone to overfitting and forgetting. As shown in Figure 1d, few-shot continual learning induces sharp topology forgetting from early stages, requiring a stability–plasticity balance to prevent the topology generator from drifting away from high-utility topology distributions.

To address these challenges, we propose MASFACT (**F**actorized **A**ligned **C**onsensus **T**opology transfer), a geometry-aware prior-to-posterior framework for continual MAS topology learning. MASFACT stores factorized topology priors, retrieves and aligns the most transferable prior to each new task via Fused Gromov–Wasserstein optimal transport (Titouan et al., 2019), and learns sparse task-specific posteriors around it. The factorized prior enables joint alignment of relational structure and agent semantics, while prior-centered posterior updates residual topology edits, balancing the task adaptation and knowledge retention under PAC-Bayes (McAllester, 1998) complexity control.

We further introduce a continual MAS evaluation protocol spanning task-, domain-, and class-incremental regimes across four reasoning domains. The protocol organizes 17 reasoning datasets into 12 continual scenarios, covering diverse topology requirements, such as verification, decomposition, execution-guided debugging, and evidence aggregation. Across all scenarios, integrating MASFACT with three representative topology generators improves average accuracy (AA) by 9.3% and reduces average forgetting (AF) by 69.4%, demonstrating robust gains over single MAS baselines.

In summary, our contributions are three-fold:

- **Problem formulation:** To our best knowledge, we are the first to formulate continual MAS topology learning and identify topology forgetting as a distinct topology-level failure mode.
- **Novel framework:** We propose MASFACT, a geometry-aware posterior transfer framework that aligns and reuses historical high-utility topologies, while constraining new-task adaptation through conservative posterior transfer.
- **Evaluation:** We design a hierarchical continual MAS evaluation protocol covering task-, domain-, and class-level task evolution and show consistent improvements on both accuracy and forgetting metrics across diverse settings.

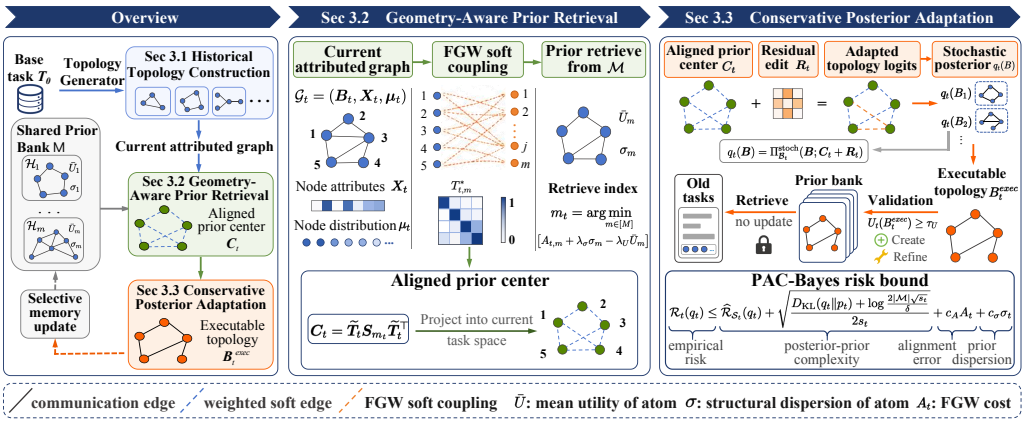


Figure 2: Overview of MASFACT. The framework has three stages: historical topology construction (Sec. 3.1), geometry-aware prior retrieval (Sec. 3.2), and conservative posterior adaptation (Sec. 3.3).

2 PROBLEM DEFINITION

Attributed MAS topology. We study MAS whose underlying LLM agent models remain fixed while their communication topology adapts across tasks. For task $\mathcal{T}_t$ with N_t agents, let $\mathcal{G}_t = (\boldsymbol{B}_t, \boldsymbol{X}_t, \boldsymbol{\mu}_t)$ be its attributed directed graph. Here, $\boldsymbol{B}_t \in \mathcal{B}_t \subseteq \{0,1\}^{N_t \times N_t}$ specifies the allowed communication links, $\boldsymbol{X}_t$ describes agent functions, available tools, and task-dependent participation conditions, and $\boldsymbol{\mu}_t$ assigns task-specific node weights that determines each agent's relative weight in cross-task topology matching. This representation separates communication structure from agent labels, allowing agents with different names to play similar structural-functional roles.

Continual MAS topology learning. We consider a task stream $\{\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_K\}$. A data-rich base task $\mathcal{T}_0$ initializes the topology generator, while each subsequent task $\mathcal{T}_t$ provides a few-shot support set $\mathcal{S}_t$ of size s_t. The underlying LLM models remain fixed, although agent pools and attributes may vary across tasks. At each stage, the generator must produce a valid topology $\boldsymbol{B}_t \in \mathcal{B}_t$ using limited new-task feedback and retained topology knowledge. Old-task evaluation is read-only, with no gradient updates or memory writes.

Learning objective. At stage t, π_t denotes the topology generator, and $\pi_t(\cdot \mid \mathcal{S}_t)$ its distribution over the feasible graphs $\mathcal{B}_t$ for task $\mathcal{T}_t$. For $\boldsymbol{B} \in \mathcal{B}_t$, define normalized support loss $\widehat{\ell}_{\mathcal{S}_t}(\boldsymbol{B}) \in [0,1]$ and utility $\widehat{U}_{\mathcal{S}_t}(\boldsymbol{B}) = 1 - \widehat{\ell}_{\mathcal{S}_t}(\boldsymbol{B})$. The expected support risk is

$$\widehat{\mathcal{R}}_{\mathcal{S}_t}(\pi_t) = \mathbb{E}_{\boldsymbol{B}\sim\pi_t(\cdot|\mathcal{S}_t)}\left[\widehat{\ell}_{\mathcal{S}_t}(\boldsymbol{B})\right]. \tag{1}$$

Let $\mathcal{R}_i(\pi_t)$ be expected loss on unseen examples from task $\mathcal{T}_i$ under read-only evaluation of the stage-t generator. When the LLM agents and old-task evaluation protocol are fixed, an increase in $\mathcal{R}_i(\pi_t)$ indicates poorer reconstruction of effective communication structures for $\mathcal{T}_i$. The goal is to minimize $\frac{1}{t+1}\sum_{i=0}^{t}\mathcal{R}_i(\pi_t)$ across seen tasks, but fitting only $\widehat{\mathcal{R}}_{\mathcal{S}_t}(\pi_t)$ can improve current-task performance while worsening earlier tasks. We therefore seek to fit new tasks while retaining effective historical communication patterns.

3 METHODOLOGY

Building on the continual MAS topology learning objective in Equation 1, we propose MASFACT, a geometry-aware posterior transfer framework for mitigating topology forgetting under few-shot task evolution. As shown in Figure 2, MASFACT converts high-utility historical topologies into transferable structural priors, aligns them to the current agent space, and performs sparse posterior adaptation under PAC-Bayes complexity control.

Specifically, MASFACT transfers historical structural knowledge through a three-stage pipeline:

- **Factorized Historical Topology Construction**: MASFACT consolidates high-utility topologies into prior atoms that retain reusable collaboration structures beyond task-specific agent identities.
- **Geometry-Aware Prior Retrieval**: Given a new task, MASFACT retrieves and aligns the most transferable prior by comparing structural relations and agent attributes.
- **Conservative Posterior Adaptation**: MASFACT learns a sparse residual posterior that edits the aligned prior, enabling adaptation while limiting structural drift.

The complete training and inference procedure of MASFACT is provided in Appendix D.8.

3.1 FACTORIZED HISTORICAL TOPOLOGY CONSTRUCTION

We represent historical structural knowledge as a bank of factorized prior atoms $\mathcal{M} = \{\mathcal{H}_m\}_{m=1}^{M}$, where each atom $\mathcal{H}_m = (\boldsymbol{S}_m, \boldsymbol{X}_m, \boldsymbol{\nu}_m)$ contains three FGW-transportable components:

- Consensus topology $\boldsymbol{S}_m$: a directed topology encoding reusable communication structure.
- Prototype attributes $\boldsymbol{X}_m$: agent roles, functions, available tools, and activation conditions.
- Node measure $\boldsymbol{\nu}_m$: normalized agent weights used in cross-task alignment.

This factorization enables topology priors to be transferred across tasks with different agent identities, role semantics, and graph cardinalities. Each atom further maintains historical utility $\bar{U}_m$ and structural dispersion σ_m as retrieval metadata. Both guide retrieval but remain outside $\mathcal{H}_m$, which defines the graph aligned by FGW. Validated high-utility patterns incompatible with existing atoms can form new atoms rather than overwrite old ones.

3.2 GEOMETRY-AWARE RETRIEVAL VIA OPTIMAL TRANSPORT

Given task $\mathcal{T}_t$, the topology generator first produces a scaffold $\mathcal{G}_t^0 = (\boldsymbol{B}_t^0, \boldsymbol{X}_t, \boldsymbol{\mu}_t)$. Its adjacency $\boldsymbol{B}_t^0$ is a coarse proposal of candidate directed communication, while $\boldsymbol{X}_t$ records current-agent attributes and $\boldsymbol{\mu}_t$ weights the agents. The scaffold guides retrieval and alignment, but is neither the graph executed by the MAS nor a hard constraint on that graph. For a historical atom $\mathcal{H}_m = (\boldsymbol{S}_m, \boldsymbol{X}_m, \boldsymbol{\nu}_m)$, differences in agent number, attributes, and ordering make direct index matching unreliable. We therefore use FGW to align agent attributes and communication relations.

FGW-based prior retrieval. We align current agents to prototype nodes through a soft coupling

$$\boldsymbol{T} \in \mathcal{U}(\boldsymbol{\mu}_t, \boldsymbol{\nu}_m) := \left\{ \boldsymbol{T} \in \mathbb{R}_+^{N_t \times N_m} \;\middle|\; \boldsymbol{T}\mathbf{1}_{N_m} = \boldsymbol{\mu}_t,\ \boldsymbol{T}^\top \mathbf{1}_{N_t} = \boldsymbol{\nu}_m \right\}, \tag{2}$$

where N_t and N_m are the numbers of current agents and prototype nodes. The entry $\boldsymbol{T}_{ij}$ assigns transport mass between current agent i and prototype node j, while the row and column constraints preserve their node measures $\boldsymbol{\mu}_t$ and $\boldsymbol{\nu}_m$.

However, matching node attributes alone is insufficient, as topology utility is fundamentally determined by its *relational structure*. We therefore adopt the *Fused Gromov-Wasserstein* distance to jointly align agent attributes and graph structures, obtaining the optimal coupling by

$$\boldsymbol{T}_{t,m}^* = \underset{\boldsymbol{T} \in \mathcal{U}(\boldsymbol{\mu}_t, \boldsymbol{\nu}_m)}{\arg\min}\ D_{\mathrm{FGW}}(\mathcal{G}_t^0, \mathcal{H}_m; \boldsymbol{T}), \tag{3}$$

where the FGW objective is

$$D_{\mathrm{FGW}}(\mathcal{G}_t^0, \mathcal{H}_m; \boldsymbol{T}) = (1-\rho)\langle \boldsymbol{C}_{t,m}^X, \boldsymbol{T} \rangle + \rho \langle \mathbf{C}_{t,m}^B, \boldsymbol{T} \otimes \boldsymbol{T} \rangle + \varepsilon \Omega(\boldsymbol{T}). \tag{4}$$

Here, ρ controls the semantic–structural trade-off. The node-attribute cost matrix $\boldsymbol{C}_{t,m}^X$ is computed from current agent attributes $\boldsymbol{X}_t$ and prototype attributes $\boldsymbol{X}_m$, with $[\boldsymbol{C}_{t,m}^X]_{ij} = c_X([\boldsymbol{X}_t]_i, [\boldsymbol{X}_m]_j)$. It measures the functional mismatch between current agent i and prototype node j. Assigning large transport mass $\boldsymbol{T}_{ij}$ to semantically incompatible nodes increases the alignment cost.

The relational cost $\mathbf{C}_{t,m}^B$ compares a candidate scaffold relation $i \to i'$ with a historical relation $j \to j'$ through $c_B([\boldsymbol{B}_t^0]_{ii'}, [\boldsymbol{S}_m]_{jj'})$. The product $\boldsymbol{T}_{ij}\boldsymbol{T}_{i'j'}$ weights this mismatch by the transport

mass assigned to both endpoint correspondences. Thus, $\boldsymbol{T} \otimes \boldsymbol{T}$ accounts for communication relations between matched agents, complementing their attribute alignment. The entropy term $\Omega(\boldsymbol{T})$ encourages soft correspondences. Full tensor definitions are provided in Appendix D.2.

For each historical atom, the optimized coupling yields the regularized alignment cost

$$A_{t,m} = D_{\mathrm{FGW}}(\mathcal{G}_t^0, \mathcal{H}_m; \boldsymbol{T}_{t,m}^*). \tag{5}$$

We favor atoms with lower alignment cost, lower structural dispersion, and higher historical utility,

$$m_t = \arg\min_{m \in [M]} \left[A_{t,m} + \lambda_\sigma \sigma_m - \lambda_U \bar{U}_m \right]. \tag{6}$$

The alignment and dispersion penalties reflect the transfer-risk analysis, while historical utility discourages selecting an easily aligned but ineffective prior. The scaffold guides selection and correspondence, while the selected atom supplies the structure for subsequent projection and residual adaptation. We write $\boldsymbol{T}_t^* := \boldsymbol{T}_{t,m_t}^*$, $A_t = A_{t,m_t}$, and $\sigma_t = \sigma_{m_t}$, keeping the selected atom and coupling fixed during posterior adaptation.

Task-space prior projection. To express the selected historical collaboration pattern over the current agents, we row-normalize the coupling and project the consensus topology,

$$\widetilde{\boldsymbol{T}}_t = \mathrm{diag}(\boldsymbol{\mu}_t)^{-1} \boldsymbol{T}_t^*, \qquad \boldsymbol{C}_t = \widetilde{\boldsymbol{T}}_t \boldsymbol{S}_{m_t} \widetilde{\boldsymbol{T}}_t^\top. \tag{7}$$

Each row of $\widetilde{\boldsymbol{T}}_t$ gives a distribution over historical prototype nodes for a current agent. Each entry $[\boldsymbol{C}_t]_{ii'}$ combines historical directed connections using the correspondence weights of current agents i and i'. The resulting $\boldsymbol{C}_t$ is a continuous topology center that parameterizes the prior distribution p_t over feasible graphs, providing the reference for subsequent residual adaptation.

3.3 PAC-Bayes-Guided Conservative Posterior Adaptation

After retrieval and alignment, the center $\boldsymbol{C}_t$ parameterizes a topology prior p_t in the current agent space. With this reference fixed, MASFACT learns a residual score matrix $\boldsymbol{R}_t$ from current-task support to obtain the posterior q_t,

$$p_t(\boldsymbol{B}) = \Pi_{\mathcal{B}_t}^{\mathrm{stoch}}(\boldsymbol{B}; \boldsymbol{C}_t), \qquad q_t(\boldsymbol{B}) = \Pi_{\mathcal{B}_t}^{\mathrm{stoch}}(\boldsymbol{B}; \boldsymbol{C}_t + \boldsymbol{R}_t). \tag{8}$$

Here, $\boldsymbol{B} \in \mathcal{B}_t$ is a feasible communication topology. The operator $\Pi_{\mathcal{B}_t}^{\mathrm{stoch}}$ maps topology scores to a distribution over feasible graphs, while $\boldsymbol{R}_t$ adjusts edge preferences relative to $\boldsymbol{C}_t$. We apply a feasibility mask to exclude invalid edges and use a straight-through Gumbel-Sigmoid estimator for differentiable sampling during training.

The transfer-risk bound below guides prior selection toward low alignment cost and dispersion, favoring compatible, internally consistent historical patterns. These quantities remain fixed during posterior adaptation, so the bound motivates balancing empirical task risk against $D_{\mathrm{KL}}(q_t \| p_t)$, which measures deviation from the selected prior. An additional residual sparsity penalty favors sparse edits, yielding the surrogate objective in Equation 10. During evaluation, MASFACT uses a high-probability topology under q_t without gradient updates or memory writing.

Theoretical guarantee. Let $\mathcal{R}_t(q_t)$ and $\widehat{\mathcal{R}}_{\mathcal{S}_t}(q_t)$ denote the expected and empirical current-task risks under q_t. We assume local risk stability around $\boldsymbol{C}_t$, such that alignment and compression of historical topologies into atoms introduce discrepancies bounded by $c_A A_t$ and $c_\sigma \sigma_t$, respectively. See Appendix D.3 for formal assumptions.

Theorem 1 (Geometry-Aware PAC-Bayes Transfer Bound)**.** *Assume the topology loss lies in* $[0, 1]$ *and the finite family of aligned candidate priors is fixed independently of the sampled support set* $\mathcal{S}_t$*. For* s_t *i.i.d. samples from the current task distribution and* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$*, the following holds simultaneously for all candidate priors* p_t *and residual posteriors* q_t *within the region covered by the stated local stability assumptions,*

$$\mathcal{R}_t(q_t) \le \widehat{\mathcal{R}}_{\mathcal{S}_t}(q_t) + \sqrt{\frac{D_{\mathrm{KL}}(q_t \| p_t) + \log \frac{2|\mathcal{M}|\sqrt{s_t}}{\delta}}{2 s_t}} + c_A A_t + c_\sigma \sigma_t. \tag{9}$$

Here, A_t *and* σ_t *correspond to the candidate prior. The logarithmic term accounts for simultaneous validity across candidates, covering the retrieved prior. The nonnegative discrepancy bounds use fixed stability constants* c_A *and* c_σ*, distinct from training weights.*

Remarks. Alignment cost and dispersion guide prior retrieval, while KL controls posterior deviation from the selected historical reference. With retrieval fixed, A_t and σ_t are constant with respect to $\boldsymbol{R}_t$. The square-root complexity term can be upper-bounded by a weighted KL term plus a constant independent of $\boldsymbol{R}_t$, motivating the following empirical objective with an additional sparsity penalty,

$$\mathcal{L}_{\text{train}} = \widehat{\mathcal{R}}_{\mathcal{S}_t}(q_t) + \lambda_{\text{KL}} D_{\text{KL}}(q_t \| p_t) + \lambda_R \|\boldsymbol{R}_t\|_1. \tag{10}$$

For fixed KL, fewer support samples increase the complexity contribution, motivating conservative adaptation. The ℓ_1 penalty encourages sparse residual edits. KL, alignment cost, and dispersion also guide whether validated evidence refines an atom or creates a new one. See Appendix D.6 for update rules. The guarantee concerns expected current-task risk, while historical forgetting is evaluated empirically. See Appendix D.4 and D.5 for the proof and surrogate derivation.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Hierarchy-aware continual MAS evaluation protocol. We instantiate the protocol along three axes of MAS evolution: task-level reasoning-paradigm shifts, domain-level difficulty and evaluation-style shifts, and class-level subskill evolution with coarse/fine splits for short-horizon adaptation and long-horizon stability. Across four reasoning families with distinct collaboration pressures, we organize 17 public datasets into 12 continual scenarios for evaluating performance, forgetting, and adaptation robustness. Details are in Appendix E.

Baselines. We compare MASFACT with four groups of baselines: single-agent methods (Vanilla, CoT, and Self-Consistency CoT (Wang et al., 2022)), fixed MAS topologies (Chain, Tree, and Complete Graph (Qian et al., 2025)), adaptive topology methods (GDesigner (Zhang et al., 2024), AgentDropout (Wang et al., 2025), ARGDesigner (Li et al., 2026), AGP (Li et al., 2025), and MasRouter (Yue et al., 2025)), and continual MAS variants with replay (Rolnick et al., 2019). We also evaluate the plug-and-play capability of MASFACT across different MAS architectures. Details are provided in Appendix F.

Implementation and Evaluation. All experiments use a unified graph-based MAS backbone following GDesigner, with fixed agent pools within each task family to isolate topology learning. We use Llama-3.1-8B-Instruct with 200 base-stage samples and 10 support samples per continual stage. MASFACT is backbone-agnostic and supports other LLMs without modifying the learning procedure. We report Average Accuracy (AA) and Average Forgetting (AF), where lower AF indicates less forgetting and negative AF denotes positive backward transfer.

4.2 MAIN RESULTS

Table 1: Class-level continual learning results. † denotes replay-based variants. Results are averaged over five random seeds. Results with standard deviations are reported in Appendix G.

Method	MMLU-Pro				MATH				TACO				2Wiki				Avg.	
	Few-shot		Standard		Few-shot		Standard		Few-shot		Standard		Few-shot		Standard		AA↑	AF↓
	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓		
Vanilla	26.17	–	–	–	35.79	–	–	–	55.15	–	–	–	39.26	–	–	–	39.09	–
CoT-ICL	39.60	4.01	–	–	36.83	4.29	–	–	58.04	4.38	–	–	50.58	2.61	–	–	46.26	3.82
SC(CoT)-ICL	40.52	4.95	–	–	37.88	6.16	–	–	59.22	3.03	–	–	58.36	2.15	–	–	49.00	4.07
Chain	36.48	–	–	–	36.92	–	–	–	58.46	–	–	–	50.62	–	–	–	45.62	–
Tree	36.82	–	–	–	36.73	–	–	–	54.55	–	–	–	52.74	–	–	–	45.21	–
Complete	38.64	–	–	–	37.63	–	–	–	59.01	–	–	–	54.68	–	–	–	47.49	–
AFLOW	40.12	3.97	42.37	4.32	42.18	3.37	48.85	3.92	62.84	3.68	64.12	4.21	60.48	2.90	61.97	3.45	52.87	3.73
AFLOW†	42.52	1.21	42.89	2.19	44.39	2.03	49.63	1.75	64.75	2.64	65.48	2.13	63.06	2.16	64.73	2.29	54.68	2.05
AgentDropout	41.07	3.13	42.48	3.83	43.54	4.18	49.88	4.96	62.72	3.17	63.14	3.53	60.07	1.81	62.84	1.97	53.22	3.32
AgentDropout†	43.03	1.72	45.84	1.49	47.05	1.56	51.75	2.42	63.38	1.70	65.91	1.42	62.94	1.53	63.58	1.10	55.44	1.62
MasRouter	45.94	2.27	46.17	2.81	52.42	3.14	54.28	4.46	67.62	3.12	68.75	3.70	65.28	2.84	67.51	3.62	58.50	3.25
MasRouter†	47.40	1.43	47.53	0.83	55.17	2.76	58.56	1.47	69.76	2.89	71.57	2.25	69.16	1.41	70.17	2.06	61.31	1.89
ARGDesigner	44.72	3.25	45.52	3.39	53.84	3.62	56.33	3.86	67.36	3.74	69.41	4.28	62.41	2.94	63.91	3.39	57.94	3.56
ARGDesigner†	46.87	1.27	48.69	1.33	54.21	1.88	57.83	1.59	68.82	2.21	70.24	1.92	65.36	1.65	67.51	2.08	59.94	1.74
GDesigner	45.04	3.28	46.72	3.61	51.92	3.89	53.36	4.95	66.41	4.81	69.42	5.26	65.09	2.74	66.27	3.63	58.03	4.02
GDesigner†	46.16	1.49	47.87	1.22	54.70	2.36	56.18	1.94	67.10	2.22	72.35	1.06	67.79	1.84	69.84	2.17	60.25	1.79
MASFACT	**49.18**	**-0.19**	**50.72**	**0.13**	**56.07**	**-0.45**	**61.68**	**0.21**	**70.94**	**0.12**	**73.80**	**-0.11**	**69.92**	**-0.26**	**71.74**	**0.08**	**62.73**	**-0.06**
Rel. Gains	↑3.76%	↓115.70%	↑4.17%	↓84.34%	↑1.63%	↓128.85%	↑5.33%	↓85.71%	↑1.69%	↓92.94%	↑2.00%	↓110.38%	↑1.10%	↓118.44%	↑2.24%	↓92.73%	↑2.32%	↓103.70%

Table 1 reports class-level continual learning results across four task families. MASFACT achieves the strongest overall performance, improving average AA while reducing average AF below zero.

Table 2: Domain- and task-level continual learning results. † denotes replay-based variants. Results are averaged over five random seeds. Results with standard deviations are reported in Appendix G.

Method	Domain-level continual learning								Task-level continual learning	
	Knowledge QA		Math		Code		Multi-hop QA		KQA→Math→Code→MQA	
	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓
Vanilla	25.57	–	29.28	–	43.78	–	35.85	–	30.84	–
CoT-ICL	34.37	5.08	30.19	6.13	46.74	3.29	38.74	4.76	34.90	5.64
SC(CoT)-ICL	32.54	7.46	33.82	6.24	50.46	3.40	41.51	3.24	37.47	5.63
Chain	29.11	–	30.63	–	45.64	–	32.03	–	32.24	–
Tree	30.40	–	34.29	–	47.12	–	37.81	–	34.17	–
Complete Graph	32.24	–	35.50	–	41.71	–	39.62	–	36.06	–
AFLOW	28.45	7.52	39.46	6.89	48.82	6.17	40.18	4.72	37.76	6.45
AFLOW†	30.21	3.89	40.29	4.16	54.77	4.22	43.68	1.89	40.39	4.12
AgentDropout	33.15	5.83	39.59	5.62	48.91	4.03	42.94	5.19	36.72	8.12
AgentDropout†	36.05	4.07	41.17	4.92	53.58	2.28	47.17	1.04	40.61	7.48
GDesigner	45.62	6.33	42.93	5.14	54.46	4.89	44.92	5.23	41.81	6.95
GDesigner†	47.73	4.20	45.52	3.57	57.13	5.04	47.97	1.26	44.21	5.15
MasRouter	46.84	4.64	43.86	4.02	58.03	4.18	45.26	3.74	45.50	7.48
MasRouter†	49.16	3.01	46.28	3.60	**62.09**	3.46	48.79	0.72	49.55	5.02
ARGDesigner	47.11	5.36	38.94	3.93	52.47	4.09	46.32	4.27	42.18	6.14
ARGDesigner†	48.70	3.16	40.40	7.59	58.41	2.76	49.32	1.90	49.13	4.83
MASFACT	**51.59**	**0.06**	**47.91**	**0.57**	61.14	**-2.03**	**52.31**	**-0.55**	**51.67**	**2.07**
Rel. Gains	↑4.94%	↓98.01%	↑3.52%	↓84.03%	↓1.53%	↓189.04%	↑6.06%	↓176.39%	↑4.28%	↓49.76%

Table 3: Fine-grained class-level continual learning results. † denotes replay-based variants.

Method	MMLU-Pro (session=13)		MATH (session=7)		TACO (session=8)		Average	
	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓
AFLOW	29.77±0.39	6.27±0.11	44.31±0.12	4.32±0.08	47.92±0.19	7.25±0.04	40.67±0.40	5.95±0.04
AFLOW†	35.10±0.46	2.78±0.13	47.76±0.27	2.92±0.04	53.41±0.21	4.36±0.09	45.42±0.13	3.35±0.08
AgentDropout	32.67±0.19	5.79±0.04	43.86±0.33	3.52±0.09	48.21±0.41	7.81±0.11	41.58±0.28	5.71±0.16
AgentDropout†	39.33±0.24	4.29±0.12	46.48±0.09	3.31±0.17	54.17±0.48	6.34±0.25	46.66±0.17	4.65±0.14
GDesigner	41.67±0.51	2.34±0.19	49.27±0.24	6.25±0.14	65.29±0.39	3.37±0.26	52.08±0.31	3.99±0.10
GDesigner†	37.95±0.64	2.93±0.13	52.39±1.18	3.01±0.14	69.72±0.56	2.89±0.17	53.35±0.84	2.94±0.12
MasRouter	40.10±0.36	4.06±0.26	54.16±0.37	5.31±0.09	63.57±0.28	4.90±0.11	52.61±0.16	4.76±0.06
MasRouter†	43.75±0.20	1.95±0.29	58.36±0.24	1.19±0.16	70.78±0.19	3.18±0.09	57.63±0.53	2.11±0.09
ARGDesigner	37.56±0.16	2.59±0.35	56.87±0.04	2.79±0.13	65.67±0.31	5.53±0.14	53.37±0.09	3.64±0.10
ARGDesigner†	45.90±0.17	1.06±0.11	54.50±0.41	3.67±0.09	69.72±0.26	3.83±0.17	56.71±0.30	2.85±0.03
MASFACT	**46.93±0.14**	**-0.78±0.27**	**58.75±0.36**	**-1.39±0.18**	**71.32±0.09**	**-0.11±0.05**	**59.00±0.18**	**-0.76±0.21**
Rel. Gains	↑2.24%	↓173.58%	↑0.67%	↓216.81%	↑0.76%	↓103.81%	↑2.38%	↓136.02%

Unlike existing topology generators and routing-based methods, which optimize each new stage around task-specific topologies and therefore retain positive AF, MASFACT converts continual adaptation into backward transfer on average, showing that previously learned collaboration patterns are not merely protected but can also benefit later evaluation.

Broader-Stream Continual Learning. Table 2 evaluates MASFACT on broader streams combining domain shifts and task transitions. MASFACT achieves the best overall AA–AF trade-off, indicating that its topology memory is not tied to a single benchmark decomposition. Although several adaptive routing or graph baselines remain competitive in AA, they still show positive forgetting after sequential adaptation. In contrast, MASFACT attains negative average AF, suggesting that transferred topology priors can improve later evaluation on previous tasks rather than merely slow degradation.

Table 3 evaluates a more demanding setting where MMLU-Pro, MATH, and TACO are decomposed into 13, 7, and 8 semantically close increments. MASFACT is the only method that consistently obtains negative AF.The table also exposes a *MAS-specific failure mode*: replay is not universally beneficial for topology generators. It can also reduce AA or increase forgetting, as seen for GDesigner on MMLU-Pro and ARGDesigner on MATH, indicating that forgetting arises not only from sample distribution shift, but also from drift in the structure-generation policy.

Replay without topology constraints may mix incompatible collaboration modes and blur fine-grained specialization. This effect becomes more evident in highly similar task increments. MASFACT

Table 4: Plug-and-play generalization of MASFACT across MAS frameworks. Green arrows report absolute gains for AA and absolute reductions for AF.

Backbone	Variant	Class-level CL				Domain-level CL				Task-level CL	
		MMLU-Pro		MATH		Code		Multi-hop QA		KQA→Math →Code→MQA	
		AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓
AGP	Original	43.49	4.37	50.80	4.03	51.89	3.92	43.05	4.29	37.22	6.70
	+ MASFACT	**48.39** ↑4.90	**1.49** ↓2.88	**54.36** ↑3.56	**0.90** ↓3.13	**57.21** ↑5.32	**2.06** ↓1.86	**49.13** ↑6.08	**1.57** ↓2.72	**40.21** ↑2.99	**3.04** ↓3.66
GDesigner	Original	45.04	3.28	51.92	3.89	54.46	4.89	44.92	5.23	41.81	6.95
	+ MASFACT	**47.93** ↑2.89	**0.79** ↓2.49	**55.98** ↑4.06	**0.68** ↓3.21	**59.14** ↑4.68	**1.43** ↓3.46	**48.32** ↑3.40	**0.23** ↓5.00	**46.53** ↑4.72	**2.49** ↓4.46
ARGDesigner	Original	44.72	3.25	53.84	3.62	52.47	4.09	46.32	4.27	42.18	6.14
	+ MASFACT	**48.45** ↑3.73	**1.21** ↓2.04	**55.93** ↑2.09	**0.52** ↓3.10	**58.76** ↑6.29	**1.05** ↓3.04	**50.53** ↑4.21	**0.31** ↓3.96	**48.44** ↑6.26	**3.32** ↓2.82

avoids this by factorizing historical structures into consensus priors and task-specific residuals, then applying FGW-constrained residual adaptation. Task-order, bank-capacity, and equal-memory replay controls are reported in Appendix G.4.

Plug-and-Play Continual Topology Transfer. We test MASFACT's generator independence by attaching its continual topology layer with three representative adaptive MAS optimizers, AGP, GDesigner, and ARGDesigner, covering pruning-based, graph-generation-based, and autoregressive graph-design paradigms. Implementation details appear in Appendix G.2. Consistent gains across all backbones under class-, domain-, and task-incremental learning (Table 4) support MASFACT as an architecture-agnostic continual topology layer. These gains indicate that structural priors provide regularization, improve topology initialization, and stabilize search, especially for dynamic optimizers otherwise prone to aggressive adaptation and drift from previously effective collaboration patterns.

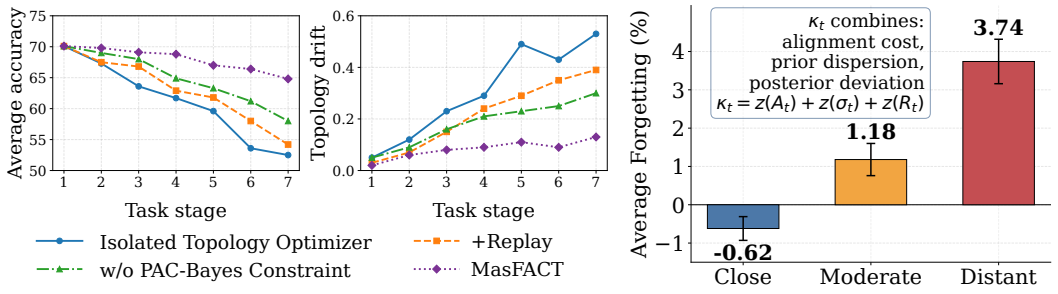
Ablation Study. Table 5 examines the contribution of each core component in MASFACT. *(1) Historical prior.* Removing the historical topology prior causes the largest degradation, demonstrating that continual MAS topology learning requires explicitly retaining reusable collaboration structures beyond current-task adaptation. *(2) Geometry-aware alignment.* The alignment ablations show that cross-task transfer requires jointly modeling topology geometry and agent semantics. Euclidean alignment underperforms the no-FGW variant due to index-wise misalignment under role or node-order drift, while GW-only alignment remains inferior to FGW because relational structure alone cannot capture agent semantics. FGW achieves reliable transfer by jointly aligning attributes and communication relations. *(3) Conservative posterior adaptation.* The residual and complexity-control ablations validate MASFACT's stability–plasticity mechanism. Removing residual editing weakens adaptation, while removing the PAC-Bayes constraint or replacing it with L2 regularization increases forgetting, confirming that MASFACT improves through prior-centered posterior control rather than generic regularization. Simpler historical reuse strategies reach 51.36–55.01 AA, below MASFACT's 59.10, indicating that storing or aligning a historical graph alone is insufficient (Appendix G.4, Table 13).

Table 5: Ablation study on the core designs under MATH 10-shot class-incremental setting.

Method	AA↑	AF↓	ΔAA	ΔAF
MASFACT	**59.10**	**-0.41**	–	–
w/o Historical Prior	54.63	2.36	-4.47	+2.77
w/o FGW Alignment	55.13	0.76	-3.97	+1.17
Euclidean Alignment	54.46	0.94	-4.64	+1.35
GW-only Alignment	56.21	0.38	-2.89	+0.79
w/o Residual Edit	55.01	0.03	-4.09	+0.44
w/o PAC-Bayes Constraint	54.71	1.27	-4.39	+1.68
L2 Complexity	56.90	0.61	-2.20	+1.02

Scaffold Sensitivity. On the MATH 10-shot setting used for the ablations, we perturb only $\boldsymbol{B}_t^0$ while fixing the support set, agent attributes, prior bank, optimizer, and adaptation procedure. With 20% edge flips, top-1 retrieval consistency remains 91.7% and AA changes from 59.10 to 58.21; a random scaffold lowers AA to 54.06 and changes AF from -0.41 to 0.96. Thus, retrieval tolerates moderate scaffold errors but still benefits from informative relational proposals (Appendix G.4, Table 14).

Mechanistic Analysis of Structural Forgetting. Figure 3a shows that old-task accuracy declines as topology drift grows. Replay partly reduces this drift. MASFACT without PAC-Bayes constraint still allows posterior updates to accumulate structural deviation despite aligned priors. In a controlled intervention that fixes the LLM agents and questions, replacing only the execution graph lowers



(a) Topology forgetting dynamics.

(b) Forgetting by prior-transfer distance.

Figure 3: Mechanistic analysis of structural forgetting. Left: topology drift tracks old-task accuracy degradation. Right: forgetting increases as the retrieved prior becomes more distant.

historical-task accuracy from 63.10 with stored topologies to 53.74 with drifted topologies. MASFACT posterior restores it to 62.58. A case study further illustrates how topology drift can impair agent collaboration. Both analyses are detailed in Appendix G (Table 15 and the topology-evolution case study). Figure 3b groups stages by the empirical transfer score κ_t, which combines alignment cost, prior dispersion, and posterior deviation. AF rises from -0.62 to 1.18 and 3.74 across close, moderate, and distant groups (Appendix G.4). This supports our transfer-complexity term in that retention benefits only from well-aligned, stable priors rather than indiscriminate reuse, explaining why MASFACT must jointly control alignment mismatch, prior instability, and posterior drift. Hyperparameter sensitivity is reported in Appendix G.1.

5 Related Work

Multi-Agent System Topology Optimization. MAS research improves task solving through agent coordination (Elrefaie et al., 2025; Zhang et al., 2025b), automated agent design (Hu et al., 2025), routing (Yue et al., 2025), and workflow or topology optimization (Li et al., 2026). These approaches typically design for collaboration structures specific task settings, leaving topology forgetting during continual adaptation unaddressed. Related graph retrieval (Bai et al., 2019) searches stored graphs by learned query similarity but leaves adaptation unspecified. Graph matching (Titouan et al., 2019) establishes node correspondences through attributes and relational structure without regulating topology changes. Although graph representation transfer (Hu et al., 2019) improves downstream prediction using pretrained representations, it does not ensure recovery of earlier collaboration structures. MASFACT instead learns task-specific posteriors around aligned historical topology priors, forming new structures through controlled changes to reusable collaboration patterns.

Continual Learning for Agents. Continual learning retains capabilities through regularization, replay, or parameter isolation (Li & Hoiem, 2017; Mallya & Lazebnik, 2018). LLM-agent research further develops capability adaptation (Zheng et al., 2025; Han et al., 2025) and memory retention (Zhong et al., 2024b). We study deployed MAS facing continual task streams with fixed agent models and adaptation confined to communication topology. Topology forgetting arises as sequential updates degrade historical topology retention. Sparse feedback in new tasks further complicates topology edits without disrupting reusable patterns. Our PAC-Bayes transfer-risk analysis therefore motivates prior-centered KL regularization of residual topology edits, balancing current-task fit against deviation from historical priors.

6 Conclusion

We identify topology forgetting as the reduced ability of a topology generator to reconstruct effective historical collaboration structures after sequential adaptation with fixed LLM agents. MASFACT addresses this challenge through aligned topology priors and constrained adaptation. We formulate prior-centered posterior learning from PAC-Bayes transfer-risk analysis, balancing new-task adaptation against deviation from reusable historical patterns. Experiments demonstrate improved accuracy and reduced forgetting across diverse continual settings with different topology generators.

AI USE STATEMENT

In this work, generative AI tools were used only for limited auxiliary tasks, including language editing and translation, reference formatting and checking. Generative AI was not used to formulate the research problem or hypotheses, develop the core theoretical or conceptual framework, design the proposed method, design or conduct experiments, implement the method, select experimental settings, generate synthetic data, or interpret empirical results. All AI-assisted text and mathematical presentation were carefully reviewed by the authors. All mathematical statements and derivations were independently checked for correctness, and references suggested or summarized by AI were manually verified against the original sources. The authors take full responsibility for the final content of the paper and for any text, claims, or artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies computational methods for large-language-model-based multi-agent systems and does not involve newly recruited human participants or the collection of new personal or sensitive data. Our experiments use existing benchmark datasets and language models under their applicable licenses or terms of use, with dataset sources, preprocessing procedures, and evaluation protocols documented in the paper and supplementary materials. As with other LLM-based multi-agent systems, practical deployments may inherit risks from the underlying models and data, including incorrect outputs, bias, privacy leakage, or misuse. Our experiments are conducted for research evaluation, and we do not advocate autonomous deployment in high-stakes settings without appropriate human oversight and system-specific safeguards. The authors have reviewed the ethical implications of this work and take responsibility for its use and presentation.

REPRODUCIBILITY STATEMENT

We have taken several measures to support the reproducibility of this work. The continual MAS topology-learning setting and objective are defined in section 2, while the prior bank, FGW-based retrieval, posterior adaptation, and training objective are specified in section 3. Formal assumptions, proofs, and the complete algorithm are provided in Appendix D. Dataset organization, task sequences, evaluation metrics, and the experimental protocol are described in section 4 and Appendix E. Baseline adaptation, implementation conventions, additional results, sensitivity analyses, and measured computational costs are documented in Appendix F, H, and G. Anonymous source code is submitted as supplementary material.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 384–392, 2019.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7889–7901, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

Mohamed Elrefaie, Janet Qian, Raina Wu, Qian Chen, Angela Dai, and Faez Ahmed. Ai agents in engineering design: a multi-agent framework for aesthetic and aerodynamic car design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 89237, pp. V03BT03A048. American Society of Mechanical Engineers, 2025.

Wenzhe Fan, Tommaso Tognoli, Henry Peng Zou, Chunyu Miao, Yibo Wang, and Xinhua Zhang. Todycomm: Task-oriented dynamic communication for multi-round llm-based multi-agent system. *arXiv preprint arXiv:2602.03688*, 2026.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.

Jiayi Han, Liang Du, Hongwei Du, Xiangguo Zhou, Yiwen Wu, Yuanfang Zhang, Weibo Zheng, and Donghong Han. Slim: Let llm learn more and forget less with soft lora and identity mixture. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4792–4804, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

Xanh Ho, Anh-Khoa Duong, and Kazunari Sugiyama. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pp. 6609–6625, 2020.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pp. 21344–21377, 2025.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Hui Yi Leong, Yuheng Li, Yuqing Wu, Wenwen Ouyang, Wei Zhu, Jiechao Gao, and Wei Han. Amas: Adaptively determining communication topology for llm-based multi-agent system. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2061–2070, 2025.

Boyi Li, Zhonghan Zhao, Der-Horng Lee, and Gaoang Wang. Adaptive graph pruning for multi-agent communication. In *ECAI 2025: 28th European Conference on Artificial Intelligence 25-30 October 2025, Bologna, Italy-Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025) Proceedings*, pp. 4305–4312. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2025.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23142–23150, 2026.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773. IEEE, 2018.

Andreas Maurer. A note on the pac bayesian theorem. *arXiv preprint cs/0411099*, 2004.

David A McAllester. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 230–234, 1998.

Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In *International Conference on Learning Representations*, volume 2025, pp. 41488–41505, 2025.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in llm-based multi-agent systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12358–12372, 2025.

Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pp. 6275–6284. PMLR, 2019.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018.

Yanwei Yue, Guibin Zhang, Boyang Liu, Guancheng Wan, Kun Wang, Dawei Cheng, and Yiyan Qi. Masrouter: Learning to route llms for multi-agent systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15549–15572, 2025.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025a.

Ling Zhang, Zijun Yao, and Arya Hadizadeh Moghaddam. Designing genai tools for personalized learning implementation: Theoretical analysis and prototype of a multi-agent system. *Journal of Teacher Education*, 76(3):280–293, 2025b.

Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8):1–35, 2025.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the association for computational linguistics: NAACL 2024*, pp. 2299–2314, 2024a.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024b.

A Appendix Overview

The appendix is structured as follows:

- Section B discusses the limitations of MASFACT and outlines future research directions.
- Section C provides the broader impact statement of this work.
- Section D presents theoretical and implementation details of MASFACT, including the stochastic topology posterior, FGW alignment, PAC-Bayes transfer bound, training objective, prior-bank update, read-only evaluation protocol, and full algorithm.
- Section E describes the hierarchical continual MAS evaluation protocol, including dataset organization, continual scenario construction, evaluation metrics, and data usage.
- Section F details the compared baselines, including single-agent methods, fixed MAS topologies, adaptive MAS topology methods, replay-enhanced variants, and plug-and-play MASFACT variants.

- Section G reports additional experiments and analyses, including hyperparameter sensitivity, plug-and-play integration details, and complete experimental results.
- Section H summarizes experimental reproducibility details, including released artifacts, evaluation protocol, few-shot settings, metrics, random seeds, hyperparameters, and compute resources.

B Limitations and Future Work

Although MASFACT improves continual MAS topology learning by explicitly preserving and transferring reusable communication structures, several limitations remain. First, the effectiveness of topology transfer depends on the quality of historical prior atoms. When early-stage topologies are noisy, low-utility, or insufficiently diverse, the prior bank may provide weak structural anchors for later tasks. Second, FGW-based alignment assumes that agent attributes and communication relations provide meaningful semantic and structural signals. In tasks where agent roles are poorly specified, highly ambiguous, or rapidly changing, the alignment cost may become less reliable.

Future work will explore uncertainty-aware prior construction, adaptive alignment objectives, and more expressive agent-attribute modeling to improve robustness under noisy or weakly structured task streams. Another promising direction is to jointly study topology continual learning and agent capability adaptation, so that communication structures and agent-level reasoning behaviors can co-evolve under explicit stability constraints. Extending the evaluation protocol to interactive environments, embodied multi-agent tasks, and real-world tool-augmented workflows would further test whether continual topology transfer remains effective under more dynamic and open-ended conditions.

C Impact Statements

Retaining useful communication patterns can reduce repeated topology search and improve coordination in reasoning systems. These benefits concern the organization of existing agent capabilities. They do not make the underlying LLM outputs reliable by themselves. Incorrect historical patterns, unsafe tool access, or misleading intermediate messages can also be reused. Deployment therefore requires controls appropriate to the tools, data, and decisions involved.

The study uses existing benchmarks and does not recruit human participants or collect new personal data. Dataset licenses and model terms continue to apply to preprocessing and redistribution. Reported token and runtime comparisons describe the evaluated configurations and do not establish a general environmental impact reduction.

D Theoretical and Implementation Details of MASFACT

This appendix provides technical details omitted from the main text. We first define the masked stochastic topology posterior used by MASFACT. We then expand the FGW alignment objective and the transport-induced projection used to push historical prior atoms into the current agent space. Next, we state the risk assumptions and prove the geometry-aware PAC-Bayes transfer bound in Theorem 1. Finally, we describe the bound-induced training surrogate, selective prior-bank update, read-only evaluation protocol, and full algorithm.

D.1 Stochastic Topology Posterior

For score matrix $\boldsymbol{Z}$, the normalized distribution over feasible directed graphs is

$$\Pi^{\mathrm{stoch}}_{\mathcal{B}_t}(\boldsymbol{B};\boldsymbol{Z}) = \frac{\exp(\langle \boldsymbol{Z}, \boldsymbol{B}\rangle)\mathbb{I}\{\boldsymbol{B}\in\mathcal{B}_t\}}{\sum_{\boldsymbol{B}'\in\mathcal{B}_t}\exp(\langle \boldsymbol{Z}, \boldsymbol{B}'\rangle)}. \tag{11}$$

The aligned center and residual scores parameterize

$$p_t(\boldsymbol{B}) = \Pi^{\mathrm{stoch}}_{\mathcal{B}_t}(\boldsymbol{B};\boldsymbol{C}_t), \qquad q_t(\boldsymbol{B}) = \Pi^{\mathrm{stoch}}_{\mathcal{B}_t}(\boldsymbol{B};\boldsymbol{C}_t+\boldsymbol{R}_t). \tag{12}$$

The center is held fixed during residual adaptation. At $\boldsymbol{R}_t = 0$, $q_t = p_t$. Residual parameterization alone does not constrain the magnitude or sparsity of changes. The KL and residual penalties provide those preferences.

Edge masks and exact factorization. Let $\mathcal{E}_t$ be the allowed ordered agent pairs. If feasibility consists only of independently masking forbidden edges, then

$$\mathcal{B}_t = \{\boldsymbol{B} \in \{0,1\}^{N_t \times N_t} : B_{ij} = 0 \text{ for } (i,j) \notin \mathcal{E}_t\},$$
$$\Pi(\boldsymbol{B}; \boldsymbol{Z}) = \prod_{(i,j)\in\mathcal{E}_t} r_{ij}^{B_{ij}} (1 - r_{ij})^{1-B_{ij}}, \qquad r_{ij} = \mathrm{sigmoid}(Z_{ij}). \tag{13}$$

Forbidden edges have probability zero. This factorization does not enforce global constraints such as connectivity, a fixed edge count, or acyclicity unless those constraints are already guaranteed by the allowed edge set. Conditioning or repairing a sampled graph changes its distribution and must be included in the probabilistic model if used.

Score convention and sampling. In the supplementary implementation, $\boldsymbol{C}_t$ is passed directly as a matrix of logits, and $\boldsymbol{C}_t + \boldsymbol{R}_t$ supplies posterior logits. Thus $[\boldsymbol{C}_t]_{ij}$ is not itself the edge probability. In particular, a zero center score gives probability $1/2$ on an allowed edge. Reinterpreting a consensus probability as a logit would change this convention and requires a corresponding implementation change. For an allowed edge with score Z_{ij}, the sampler draws u_{ij} uniformly on $(0,1)$ and computes

$$y_{ij} = \mathrm{sigmoid}\left(\frac{Z_{ij} + \log u_{ij} - \log(1 - u_{ij})}{\tau}\right), \qquad B_{ij} = \mathbb{I}\{y_{ij} \geq 1/2\}. \tag{14}$$

For $\tau > 0$, the hard draw has Bernoulli probability $\mathrm{sigmoid}(Z_{ij})$. The straight-through estimator uses the hard value in the forward pass and the derivative of y_{ij} in the backward pass. This gradient estimator is biased, although the hard forward distribution is the stated Bernoulli law at threshold $1/2$. Other thresholds require a corresponding adjustment of the sampling law.

Graph KL and training normalization. Writing $p_{ij} = \mathrm{sigmoid}(C_{t,ij})$ and $q_{ij} = \mathrm{sigmoid}(C_{t,ij} + R_{t,ij})$, the exact KL for the factorized graph distributions is

$$K_t := D_{\mathrm{KL}}(q_t \| p_t) = \sum_{(i,j)\in\mathcal{E}_t} \left[q_{ij} \log \frac{q_{ij}}{p_{ij}} + (1 - q_{ij}) \log \frac{1 - q_{ij}}{1 - p_{ij}} \right]. \tag{15}$$

The supplementary training routine returns K_t/d_t and $\|\boldsymbol{R}_t\|_{1,\mathcal{E}_t}/d_t$, where $d_t = \max(1, |\mathcal{E}_t|)$. These are average penalties per allowed edge. The PAC-Bayes bound uses the full graph KL K_t, not its average. Training weights can absorb this normalization for a fixed d_t, but the conversion must be made explicit when graph sizes differ. For coupled graph constraints, Equation 15 is no longer the KL of Equation 11 without a separate derivation.

Execution. For independent edges, a maximum-probability execution graph is obtained by thresholding posterior probabilities at $1/2$, with fixed tie breaking,

$$\boldsymbol{B}_t^{\mathrm{exec}} \in \arg\max_{\boldsymbol{B}\in\mathcal{B}_t} q_t(\boldsymbol{B}). \tag{16}$$

Evaluation performs no gradient updates or memory writes. A bound on $\mathbb{E}_{\boldsymbol{B}\sim q_t} \ell_t(\boldsymbol{B}; z)$ does not automatically bound the loss of this single selected graph.

D.2 FGW Alignment and Transport-Induced Projection

The scaffold $\mathcal{G}_t^0 = (\boldsymbol{B}_t^0, \boldsymbol{X}_t, \boldsymbol{\mu}_t)$ supplies a coarse proposal of current communication relations. Agent attributes separately encode roles, functions, resources, and participation conditions. For atom $\mathcal{H}_m = (\boldsymbol{S}_m, \boldsymbol{X}_m, \boldsymbol{\nu}_m)$, normalized node measures define

$$\mathcal{U}(\boldsymbol{\mu}_t, \boldsymbol{\nu}_m) = \{\boldsymbol{T} \in \mathbb{R}_+^{N_t \times N_m} : \boldsymbol{T}\mathbf{1}_{N_m} = \boldsymbol{\mu}_t,\ \boldsymbol{T}^\top \mathbf{1}_{N_t} = \boldsymbol{\nu}_m\}. \tag{17}$$

The supplementary implementation normalizes nonnegative supplied weights and uses uniform weights when none are supplied. Zero-mass nodes are excluded from inverses used below. Coupling entries are transport masses, whereas normalized rows give conditional correspondence weights.

FGW discrepancy. The objective is

$$D_{\mathrm{FGW}}(\mathcal{G}_t^0, \mathcal{H}_m; \boldsymbol{T}) = (1-\rho)\langle \boldsymbol{C}_{t,m}^X, \boldsymbol{T}\rangle + \rho\langle \mathbf{C}_{t,m}^B, \boldsymbol{T}\otimes\boldsymbol{T}\rangle + \varepsilon\Omega(\boldsymbol{T}), \tag{18}$$

where $\rho \in [0,1]$ balances attributes and relations and $\varepsilon \geq 0$ controls entropy regularization. The costs are

$$[\boldsymbol{C}_{t,m}^X]_{ij} = c_X([\boldsymbol{X}_t]_i, [\boldsymbol{X}_m]_j), \tag{19}$$

$$[\mathbf{C}_{t,m}^B]_{(i,i'),(j,j')} = c_B([\boldsymbol{B}_t^0]_{ii'}, [\boldsymbol{S}_m]_{jj'}), \tag{20}$$

$$[\boldsymbol{T}\otimes\boldsymbol{T}]_{(i,i'),(j,j')} = \boldsymbol{T}_{ij}\boldsymbol{T}_{i'j'}, \tag{21}$$

$$\Omega(\boldsymbol{T}) = \sum_{ij} \boldsymbol{T}_{ij}(\log \boldsymbol{T}_{ij} - 1), \tag{22}$$

with $0\log 0 = 0$. The implementation defaults to cosine dissimilarity for attributes and squared difference for scalar edge values. The attribute encoder and cost scales must be kept consistent across current and prototype agents. Matching both endpoints compares the directed relation $i \to i'$ with $j \to j'$.

Regularized score and nonnegative distortion. The ideal coupling and retrieval score are

$$\boldsymbol{T}_{t,m}^* \in \operatorname*{arg\,min}_{\boldsymbol{T}\in\mathcal{U}(\boldsymbol{\mu}_t,\boldsymbol{\nu}_m)} D_{\mathrm{FGW}}(\mathcal{G}_t^0, \mathcal{H}_m; \boldsymbol{T}), \tag{23}$$

$$A_{t,m} = D_{\mathrm{FGW}}(\mathcal{G}_t^0, \mathcal{H}_m; \boldsymbol{T}_{t,m}^*). \tag{24}$$

This regularized score can be negative because Ω can be negative. For risk analysis, distinguish the nonnegative attribute and relation discrepancy

$$\Delta_{t,m} = (1-\rho)\langle \boldsymbol{C}_{t,m}^X, \boldsymbol{T}_{t,m}^*\rangle + \rho\langle \mathbf{C}_{t,m}^B, \boldsymbol{T}_{t,m}^*\otimes\boldsymbol{T}_{t,m}^*\rangle \geq 0. \tag{25}$$

Thus $A_{t,m} = \Delta_{t,m} + \varepsilon\Omega(\boldsymbol{T}_{t,m}^*)$. A nonnegative transfer-risk allowance uses $\Delta_{t,m}$ or a separately justified nonnegative discrepancy, not the raw regularized score. This distinction leaves the stated retrieval ranking unchanged and avoids silently altering the reported experiments. It must also be respected when interpreting the main-text risk correction.

FGW optimization is nonconvex. The star denotes the ideal optimum in the mathematical definition, while finite numerical iterations provide an approximate coupling. For directed relations, the derivative of the structural term includes contributions from both endpoints,

$$\frac{\partial}{\partial T_{ab}} \sum_{i,i',j,j'} C^B_{(i,i'),(j,j')} T_{ij} T_{i'j'} = \sum_{i',j'} C^B_{(a,i'),(b,j')} T_{i'j'} + \sum_{i,j} C^B_{(i,a),(j,b)} T_{ij}. \tag{26}$$

Twice the first sum is valid only when the relevant symmetry holds. Numerical solver settings and marginal residuals are separate from claims of a global optimum.

Projection and scaffold influence. For the selected atom, write $\boldsymbol{T}_t^* = \boldsymbol{T}_{t,m_t}^*$ and normalize

$$\widetilde{\boldsymbol{T}}_t = \mathrm{diag}(\boldsymbol{\mu}_t)^{-1}\boldsymbol{T}_t^*. \tag{27}$$

The forward projection is

$$\boldsymbol{C}_t = \widetilde{\boldsymbol{T}}_t \boldsymbol{S}_{m_t} \widetilde{\boldsymbol{T}}_t^\top, \qquad [C_t]_{ii'} = \sum_{j,j'} [\widetilde{T}_t]_{ij} [S_{m_t}]_{jj'} [\widetilde{T}_t]_{i'j'}. \tag{28}$$

Each current connection combines historical directed connections using both endpoint correspondences. The scaffold affects atom selection and correspondence, while the historical consensus supplies the projected structure. Residual learning can subsequently increase or decrease edge probabilities. This mechanism does not guarantee correction of every scaffold error. Retrieval also uses historical utility and dispersion, and its empirical sensitivity is examined in Appendix G.4.

For memory updates, the reverse normalized map is

$$\boldsymbol{Q}_t = \mathrm{diag}(\boldsymbol{\nu}_{m_t})^{-1}(\boldsymbol{T}_t^*)^\top. \tag{29}$$

It maps current posterior statistics back to prototype nodes and is not generally an inverse of the forward projection.

D.3 Risk Definitions and Assumptions

Let $\mathcal{D}_t$ be the current task distribution and $\mathcal{S}_t = (z_1, \ldots, z_{s_t})$ its support sample. For normalized topology loss $\ell_t(\boldsymbol{B}; z) \in [0, 1]$, define

$$\mathcal{R}_t(q) = \mathbb{E}_{\boldsymbol{B}\sim q}\mathbb{E}_{z\sim\mathcal{D}_t}\ell_t(\boldsymbol{B}; z), \qquad \widehat{\mathcal{R}}_{\mathcal{S}_t}(q) = \mathbb{E}_{\boldsymbol{B}\sim q}\frac{1}{s_t}\sum_{r=1}^{s_t}\ell_t(\boldsymbol{B}; z_r). \tag{30}$$

LLM execution randomness is included in the loss expectation under a fixed execution protocol. Empirical estimates of these expectations have their own Monte Carlo error. The analysis is for the current task at a fixed stage.

Bounded loss and sampling. Utility is normalized as $U = 1 - \ell$ before applying the bounded-loss result. Percentage scores are divided by 100 for this purpose. Conditional on past tasks and permitted task metadata, the support examples are assumed independent with common law $\mathcal{D}_t$. This condition is stronger than merely observing a sequential benchmark.

Prior construction and prior selection. For a fixed stage, write $M = |\mathcal{M}|$ for the bank size before posterior fitting. The finite candidate family $\{p_{t,m}\}_{m=1}^{M}$ must be fixed independently of the sampled support set used in Equation 30. After this family is fixed, its selected index and the posterior may depend on that support sample because the bound holds simultaneously over the family. A historical bank alone does not ensure that the *aligned* candidates are sample independent. If scaffold construction, attribute estimation, or FGW alignment uses those same examples, a separate argument is needed.

The plug-in protocol in Appendix G.2 permits a scaffold constructed from current support. Its results therefore do not by themselves instantiate the sample-independent-prior theorem. A valid alternative is to construct candidates from independently sampled alignment data or independently specified task information, and evaluate the bound on a separate support sample. Another alternative is a theorem for data-dependent prior construction.

Local FGW stability. A local alignment assumption must identify the risk functionals being compared, the posterior family over which it holds, and a nonnegative discrepancy such as $\Delta_t = \Delta_{t,m_t}$. A bound of the form $c_A\Delta_t$ is an assumed control of a specified risk difference. Small FGW cost alone does not prove small changes in LLM execution risk. In particular, different directed paths can change verification or aggregation discontinuously. The local condition is restricted to the stated adaptation family and does not imply global continuity over arbitrary graph edits.

Prior compression stability. Dispersion σ_t measures variation among structures summarized by an atom. A risk correction $c_\sigma\sigma_t$ additionally assumes that this structural variation controls a specified change in expected task loss. The statistic and the risk-sensitivity constant are distinct. The empirical memory rule does not estimate a universally valid stability constant. The next subsection distinguishes the standard PAC-Bayes result, a conservative augmented bound, and a conditional reference-risk construction.

D.4 Proof of the Geometry-Aware PAC-Bayes Transfer Bound

Fixed-prior PAC-Bayes bound. Under the bounded-loss and sample-independence conditions above, the PAC-Bayes binary-kl result gives (Maurer, 2004), with probability at least $1 - \delta$, simultaneously for all graph posteriors q,

$$\mathcal{R}_t(q) \le \widehat{\mathcal{R}}_{\mathcal{S}_t}(q) + \sqrt{\frac{D_{\mathrm{KL}}(q\|p) + \log(2\sqrt{s_t}/\delta)}{2s_t}}. \tag{31}$$

For completeness, let $\widehat{\ell}_{\mathcal{S}_t}(\boldsymbol{B})$ and $\ell_{\mathcal{D}_t}(\boldsymbol{B})$ denote empirical and population loss for a fixed graph. The exponential moment of $s_t\,\mathrm{kl}(\widehat{\ell}_{\mathcal{S}_t}(\boldsymbol{B})\|\ell_{\mathcal{D}_t}(\boldsymbol{B}))$ is at most $2\sqrt{s_t}$. The cited result proves this for $s_t \ge 8$. For $1 \le s_t \le 7$, the exact Bernoulli moment $\sum_{k=0}^{s_t}\binom{s_t}{k}(k/s_t)^k(1-k/s_t)^{s_t-k}$ satisfies the same inequality, with $0^0 = 1$. Integrating over a sample-independent prior and applying Markov's

inequality bounds the log moment with probability $1-\delta$. The variational KL inequality and convexity of binary KL then give

$$s_t \,\mathrm{kl}(\widehat{\mathcal{R}}_{\mathcal{S}_t}(q)\|\mathcal{R}_t(q)) \le D_{\mathrm{KL}}(q\|p) + \log(2\sqrt{s_t}/\delta) \tag{32}$$

simultaneously for every q. One obtains Equation 31 using the inequality $\mathrm{kl}(a\|b) \ge 2(a-b)^2$. Here kl is the KL between Bernoulli distributions with means a, b, and the graph KL uses Equation 15 when edges factorize. Infinite KL gives a vacuous bound. No local geometry assumption is needed for Equation 31 itself.

Finite candidate selection. Apply Equation 31 to each sample-independent candidate $p_{t,m}$ with failure probability δ/M. A union bound gives simultaneous validity for all m and q,

$$\mathcal{R}_t(q) \le \widehat{\mathcal{R}}_{\mathcal{S}_t}(q) + \sqrt{\frac{D_{\mathrm{KL}}(q\|p_{t,m}) + \log(2M\sqrt{s_t}/\delta)}{2s_t}}. \tag{33}$$

It therefore covers the retrieved index m_t even if selection uses support feedback. The argument covers selection from a fixed family, not support-dependent construction of that family. A guarantee across multiple continual stages would additionally allocate failure probabilities across stages.

What the geometric allowances establish. For nonnegative $c_A\Delta_t$ and $c_\sigma\sigma_t$, Equation 33 also implies the conservative bound

$$\mathcal{R}_t(q_t) \le \widehat{\mathcal{R}}_{\mathcal{S}_t}(q_t) + \sqrt{\frac{D_{\mathrm{KL}}(q_t\|p_t) + \log(2M\sqrt{s_t}/\delta)}{2s_t}} + c_A\Delta_t + c_\sigma\sigma_t. \tag{34}$$

This is the valid conservative interpretation of the augmented form in Theorem 1, provided its alignment correction denotes the nonnegative distortion Δ_t. Adding nonnegative allowances makes an existing bound looser and does not prove that alignment and compression must appear in the current-task generalization gap. The regularized retrieval score A_t in Equation 24 cannot be substituted for Δ_t without an additional justification.

A conditional risk-transfer decomposition. A stronger transfer interpretation requires a separately defined reference loss $\widetilde{\ell}_{t,m}(\boldsymbol{B}; z) \in [0,1]$, fixed independently of the risk-estimation sample. Let $\widetilde{\mathcal{R}}_{t,m}(q)$ and $\widehat{\widetilde{\mathcal{R}}}_{\mathcal{S}_t,m}(q)$ be its population and empirical risks. If a stated local stability assumption gives

$$\mathcal{R}_t(q) \le \widetilde{\mathcal{R}}_{t,m}(q) + c_A\Delta_{t,m} + c_\sigma\sigma_m, \tag{35}$$

then applying the finite-family PAC-Bayes bound to that reference loss proves

$$\mathcal{R}_t(q) \le \widehat{\widetilde{\mathcal{R}}}_{\mathcal{S}_t,m}(q) + \sqrt{\frac{D_{\mathrm{KL}}(q\|p_{t,m}) + \log(2M\sqrt{s_t}/\delta)}{2s_t}} + c_A\Delta_{t,m} + c_\sigma\sigma_m. \tag{36}$$

The empirical term here is the empirical *reference* loss.

Scope. The proved fixed-prior result concerns the expected loss of a stochastic topology on the current task. Empirical interventions and retention measurements address those questions separately. Large graph KL can also make the bound exceed one, in which case it gives no useful numerical risk certificate.

D.5 Bound-Induced Training Objective

With the retrieved prior fixed, let $K_t = D_{\mathrm{KL}}(q_t\|p_t)$ and $b_t = \log(2M\sqrt{s_t}/\delta)$. For every fixed $\lambda > 0$, the inequality $2ab \le a^2 + b^2$ gives

$$\sqrt{\frac{K_t + b_t}{2s_t}} \le \lambda K_t + \lambda b_t + \frac{1}{8s_t\lambda}. \tag{37}$$

The last two terms do not depend on residual parameters. Thus a tunable KL penalty gives a valid upper-bound surrogate for the complexity term under the theorem's conditions. It is not a unique

optimization rule or a claim that a numerically tight bound is minimized in every experiment. Adding the separate sparsity preference gives

$$\mathcal{L}_{\text{train}} = \widehat{\mathcal{R}}_{\mathcal{S}_t}(q_t) + \lambda_{\text{KL}} D_{\text{KL}}(q_t \| p_t) + \lambda_R \|\boldsymbol{R}_t\|_1. \tag{38}$$

Retrieval determines the atom and coupling. Its alignment and dispersion quantities are fixed additive terms during residual optimization and contribute no residual gradients, although the selected prior still affects gradients through KL. Historical utility further favors effective priors and is not a PAC-Bayes term. The residual ℓ_1 term encourages sparse score changes and does not guarantee that every changed edge is necessary.

For the supplementary routine's average penalties, a code weight $\lambda_{\text{KL}}^{\text{code}}$ corresponds to the full-KL weight $\lambda_{\text{KL}}^{\text{code}}/d_t$, and similarly for residual ℓ_1. This conversion is essential for comparing graph sizes and for evaluating a theorem using full KL. For fixed KL, fewer support samples enlarge the complexity contribution. The theory therefore motivates conservative adaptation without proving a universal stability or forgetting guarantee.

D.6 Selective Prior-Bank Update

After adaptation, the bank is updated only with evidence that passes a utility threshold. Let $\boldsymbol{B}_t^{\text{exec}}$ be the selected execution graph and

$$\overline{\boldsymbol{B}}_t = \mathbb{E}_{\boldsymbol{B}\sim q_t}[\boldsymbol{B}] \tag{39}$$

its posterior mean adjacency. The former measures execution utility, while the latter supplies structural evidence. Under independent edges, the posterior mean consists of edge probabilities. The documented gate evaluates $\widehat{U}_{\mathcal{S}_t}(\boldsymbol{B}_t^{\text{exec}})$ on support. Passing this gate is not an independent validation guarantee.

Selection, refinement, and expansion. The decision score combines posterior deviation, alignment, and dispersion,

$$\kappa_t = D_{\text{KL}}(q_t \| p_t) + c_A A_t + c_\sigma \sigma_t. \tag{40}$$

This is a memory decision score inspired by the analysis, not the PAC-Bayes bound itself. The KL term is not under the theorem's square root, the selection term is absent, and the regularized A_t may be signed. Using coefficients in this decision rule does not identify the unknown constants of a risk stability assumption. For nonnegative geometric diagnostics, Δ_t in Equation 25 must replace the raw score with a correspondingly calibrated threshold.

If utility is below τ_U, the posterior does not update the bank. Otherwise, a score at most τ_κ refines the selected atom, while a larger score creates an atom. This permits useful new collaboration patterns to be retained instead of forcing every task into an existing mode. High score alone never warrants expansion without sufficient utility.

Refinement equations. The reverse correspondence transfers the posterior summary into prototype coordinates,

$$\widetilde{\boldsymbol{S}}_t = \boldsymbol{Q}_t \overline{\boldsymbol{B}}_t \boldsymbol{Q}_t^\top. \tag{41}$$

The updated consensus and utility are

$$\boldsymbol{S}_{m_t}^{\text{new}} = (1-\eta_t)\boldsymbol{S}_{m_t}^{\text{old}} + \eta_t \widetilde{\boldsymbol{S}}_t, \qquad \bar{U}_{m_t}^{\text{new}} = (1-\eta_t)\bar{U}_{m_t}^{\text{old}} + \eta_t \widehat{U}_{\mathcal{S}_t}(\boldsymbol{B}_t^{\text{exec}}). \tag{42}$$

The supplementary implementation updates dispersion by mean absolute deviation from the updated consensus,

$$\sigma_{m_t}^{\text{new}} = (1-\eta_t)\sigma_{m_t}^{\text{old}} + \eta_t \frac{\|\widetilde{\boldsymbol{S}}_t - \boldsymbol{S}_{m_t}^{\text{new}}\|_1}{N_{m_t}^2}. \tag{43}$$

This statistic is nonnegative and summarizes structural consistency. It is not an FGW distance or a measured task-risk uncertainty. Prototype attributes and node measures stay in their existing coordinates in this update.

Expansion and finite capacity. A new atom is initialized in the current agent space,

$$\mathcal{H}_{\text{new}} = (\overline{\boldsymbol{B}}_t, \boldsymbol{X}_t, \boldsymbol{\mu}_t), \qquad \bar{U}_{\text{new}} = \widehat{U}_{\mathcal{S}_t}(\boldsymbol{B}_t^{\text{exec}}), \qquad \sigma_{\text{new}} = 0. \tag{44}$$

Zero initial dispersion reflects one observation, not demonstrated reliability. When a finite capacity is set, the supplementary implementation ranks atoms by utility minus dispersion and retains the prescribed number. Its unbounded default supplies no worst-case growth guarantee. The observed growth from 13 to 19 atoms and the capacity sweep are reported in Appendix G.4.

Implementation distinction. The supplementary wrapper accepts an explicitly supplied transfer score. If none is supplied, its fallback uses the mean absolute difference between prior and posterior edge probabilities, together with alignment and dispersion. That fallback is not graph KL. Experiments using Equation 40 require the explicit score with the intended KL normalization.

D.7 Read-only Evaluation Protocol

After each stage, current and earlier tasks are evaluated with the stage's learned topology mechanism. The LLM weights and each evaluated task's agent definitions, prompts, inputs, aggregation rule, and execution budget are fixed. Retrieval and alignment may be recomputed, but posterior parameters are not optimized from old-task test examples, test labels do not enter retrieval or gating, and no bank write occurs. Any trained mapping used to produce residual scores is frozen during evaluation. Storing separate task residuals, if used, constitutes additional persistent state and must be included in memory accounting.

This protocol measures reconstruction of useful historical communication rather than relearning from old-task labels. It does not require the generated graph to be identical to its earlier version. A different topology may preserve or improve task utility. The topology-only intervention in Appendix G.4 separately replaces execution graphs while fixing all other inputs to test their causal effect.

D.8 Algorithm Details

Algorithm 1 summarizes the topology operations. The base stage supplies high-utility historical graphs and the initial atoms. The reported MMLU-Pro configuration initializes 13 atoms from 200 base examples. This count is an empirical initialization outcome, not a universal one-atom-per-example or one-atom-per-task rule. The supplementary core exposes atom initialization.

Why the components are separate. The retained object is an executable collaboration pattern. Retrieval chooses a candidate, FGW establishes correspondence across agent spaces, projection transfers its relations, and residual learning adapts its graph distribution. Graph similarity alone does not specify this adaptation, and correspondence alone does not constrain later changes. The memory gate supports both refinement and expansion. This is the intended contribution of the combined design, without claiming novelty in FGW or PAC-Bayes individually. The comparison with simpler reuse methods appears in Appendix G.4.

E Details of the Hierarchical Continual MAS Evaluation Protocol

This appendix provides details of the datasets and task construction used in our hierarchical continual MAS evaluation protocol. The goal of this protocol is not to introduce new raw datasets, but to reorganize existing reasoning benchmarks into continual MAS learning scenarios. The protocol is designed to evaluate whether a MAS can maintain strong system-level performance when task requirements, data distributions, reasoning styles, and subskills evolve over time.

E.1 Overview of Dataset Organization

We organize 17 datasets into four reasoning families: knowledge QA, mathematical reasoning, code generation, and knowledge-intensive multi-hop reasoning. These families are selected because they induce different collaboration requirements among agents. Knowledge QA emphasizes answer

Algorithm 1 MASFACT training and evaluation

Require: Task stream, support sets, fixed LLM agents, topology optimizer, utility and score thresholds
Ensure: Updated topology mechanism and bank $\mathcal{M}$

```
Learn the base-stage topology mechanism and initialize atoms from useful historical patterns
for each subsequent task T_t do
    Construct scaffold G_t^0 and record the information used for its construction
    for each stored atom H_m do
        Compute a soft coupling and retrieval score by Eqs. 23 and 24
    end for
    Select an atom using alignment, dispersion, and historical utility
    Project its consensus into C_t and hold the resulting p_t fixed
    Learn residual scores with the objective in Equation 38
    Evaluate execution utility on support and compute the specified memory score
    if utility is below τ_U then
        Keep the bank unchanged
    else if memory score is at most τ_κ then
        Refine the selected atom using posterior statistics
    else
        Initialize an atom from the useful new pattern
    end if
    Enforce the configured capacity, if finite
    Evaluate seen tasks with frozen parameters, no test-label feedback, and no memory writes
end for
```

verification and conflict resolution; mathematical reasoning emphasizes decomposition and step-wise checking; code generation emphasizes generation, execution, debugging, and repair; and multi-hop reasoning emphasizes evidence retrieval and aggregation.

Based on these datasets, we construct continual learning scenarios at three levels of task evolution:

- **Task-level evolution:** stages correspond to different reasoning families, forming a cross-scenario continual learning sequence.
- **Domain-level evolution:** stages correspond to different datasets within the same reasoning family, capturing distribution, difficulty, and evaluation-style shifts.
- **Class-level evolution:** stages correspond to coarse- or fine-grained subcategories inside a representative benchmark, capturing subskill evolution and long-horizon continual learning.

This design yields 12 continual MAS scenarios: one task-level sequence, four domain-level sequences, four coarse-grained class-level sequences, and three fine-grained class-level sequences.

E.2 KNOWLEDGE QA DATASETS

MMLU. MMLU (Hendrycks et al., 2020) is a broad multi-task knowledge QA benchmark covering subjects from humanities, social sciences, STEM, and professional domains. We use MMLU as one stage in the knowledge QA domain-incremental sequence. It represents a broad but relatively standard multiple-choice knowledge reasoning setting.

MMLU-Pro. MMLU-Pro (Wang et al., 2024) is a more challenging extension of MMLU, with more demanding multiple-choice questions. In our protocol, MMLU-Pro serves two roles. First, it is used as one stage in the knowledge QA domain-level sequence. Second, because of its rich category structure, it is used to construct both coarse-grained and fine-grained class-incremental settings. Specifically, we split MMLU-Pro into 13 fine-grained classes and 3 coarse-grained groups. The fine-grained split is used to test long-horizon continual learning, while the coarse-grained split is used to test shorter-horizon adaptation across broader knowledge categories.

AGIEval. AGIEval (Zhong et al., 2024a) contains questions derived from standardized and professional examinations, making it suitable for evaluating advanced reasoning and exam-style problem solving. We use AGIEval as one stage in the knowledge QA domain-incremental sequence. Compared with MMLU-style evaluation, AGIEval introduces a different distribution and evaluation style.

GPQA. GPQA (Rein et al., 2023) is a difficult graduate-level question answering benchmark designed to test expert-level scientific reasoning. We use GPQA as the most challenging stage in the knowledge QA domain-level sequence. Its inclusion increases the difficulty gradient and tests whether the MAS can reorganize collaboration for more expert-level knowledge reasoning.

E.3 Mathematical Reasoning Datasets

GSM8K. GSM8K (Cobbe et al., 2021) is a grade-school mathematical reasoning dataset consisting mainly of arithmetic word problems. In our protocol, GSM8K is used as an early stage in the mathematics domain-level sequence. It represents relatively short, natural-language mathematical reasoning with clear numeric answers.

MATH. MATH (Hendrycks et al., 2021) contains competition-style mathematical problems across multiple subjects, including algebra, geometry, number theory, counting and probability, intermediate algebra, prealgebra, and precalculus. We use MATH both as one stage in the mathematics domain-level sequence and as the representative benchmark for class-level mathematical continual learning. We construct 7 fine-grained class stages based on its native subject categories and additionally group them into 3 coarse-grained stages for shorter-horizon class-incremental evaluation.

Olympiad-level Mathematics Dataset. Olympiad (He et al., 2024) contains problems requiring stronger abstraction, multi-step derivation, and proof-like reasoning. It is used to test whether the MAS can adapt its collaboration structure from ordinary mathematical word problems to more challenging competition-style reasoning.

TheoremQA. TheoremQA (Chen et al., 2023) evaluates theorem-grounded mathematical and scientific reasoning. Compared with standard problem-solving datasets, TheoremQA requires the system to identify and apply relevant theorems or formal principles. We use it as one stage in the mathematics domain-level sequence to introduce a shift from numerical problem solving to theorem-based reasoning.

E.4 Code Generation Datasets

APPS. APPS (Hendrycks et al., 2021) is a programming problem dataset containing natural-language problem statements and corresponding coding solutions. We use APPS as one stage in the code domain-level sequence. It emphasizes algorithmic problem solving and program synthesis under natural-language specifications.

HumanEval. HumanEval (Chen et al., 2021) is a code generation benchmark consisting of function-level programming tasks with unit tests. We use HumanEval as one stage in the code domain-level sequence. Compared with APPS, it focuses more on concise function completion and test-based correctness.

LiveCodeBench. LiveCodeBench (Jain et al., 2025) is a code generation benchmark designed to evaluate contemporary coding ability under more realistic and temporally updated programming tasks. We use it as one stage in the code domain-level sequence to introduce a distribution shift toward more recent and challenging code-generation problems.

MBPP. MBPP (Austin et al., 2021) consists of mostly entry-level Python programming problems. We use MBPP as one stage in the code domain-level sequence. It provides a complementary programming distribution with relatively short problem statements and clear functional requirements.

TACO. TACO (Li et al., 2023) is used as the representative benchmark for code class-incremental evaluation. Because it contains diverse programming categories, we split TACO into 8 fine-grained classes and 3 coarse-grained groups. The fine-grained setting tests long-horizon code-task continual learning, while the coarse-grained setting evaluates broader subskill shifts in code generation.

E.5 Knowledge-Intensive Multi-hop Reasoning Datasets

HotpotQA. HotpotQA (Yang et al., 2018) is a multi-hop QA dataset requiring reasoning over multiple pieces of evidence. We use HotpotQA as one stage in the multi-hop QA domain-level sequence. It emphasizes evidence retrieval, intermediate reasoning, and answer aggregation.

2WikiMultihopQA. 2WikiMultihopQA (Ho et al., 2020) contains multi-hop questions grounded in Wikipedia-style evidence. In our protocol, 2Wiki serves two roles. First, it is used as one stage in the multi-hop QA domain-level sequence. Second, because it contains structured reasoning categories, we use it to construct a 4-class class-incremental setting for multi-hop reasoning.

MuSiQue. MuSiQue (Trivedi et al., 2022) is a multi-hop QA dataset designed to reduce shortcut reasoning and require more compositional evidence aggregation. We use MuSiQue as one stage in the multi-hop QA domain-level sequence. It introduces a stronger requirement for multi-step evidence integration.

StrategyQA. StrategyQA (Geva et al., 2021) contains questions that require implicit multi-step reasoning and strategic inference. We use StrategyQA as one stage in the multi-hop reasoning domain-level sequence. Its inclusion introduces a shift from explicit evidence-chain multi-hop QA toward implicit reasoning and question decomposition.

E.6 Construction of Continual Scenarios

Task-level continual learning. The task-level setting contains a four-stage sequence across reasoning families:

$$\text{Knowledge QA} \rightarrow \text{Mathematics} \rightarrow \text{Code Generation} \rightarrow \text{Multi-hop Reasoning}. \tag{45}$$

This setting evaluates whether a MAS can adapt across distinct reasoning paradigms while preserving performance on previously seen task families.

Domain-level continual learning. For each reasoning family, we construct a domain-level sequence using datasets with different distributions, difficulty profiles, or evaluation styles:

- **Knowledge QA:** MMLU $\rightarrow$ MMLU-Pro $\rightarrow$ AGIEval $\rightarrow$ GPQA.
- **Mathematics:** GSM8K $\rightarrow$ MATH $\rightarrow$ Olympiad-level mathematics $\rightarrow$ TheoremQA.
- **Code generation:** APPS $\rightarrow$ HumanEval $\rightarrow$ LiveCodeBench $\rightarrow$ MBPP.
- **Multi-hop reasoning:** HotpotQA $\rightarrow$ 2WikiMultihopQA $\rightarrow$ MuSiQue $\rightarrow$ StrategyQA.

These sequences evaluate MAS performance under meso-level shifts in data distribution, task difficulty, and evaluation style within the same reasoning family.

Class-level continual learning. For representative datasets with rich internal category structures, we construct class-level continual learning settings. MMLU-Pro, MATH, TACO, and 2Wiki are used for coarse-grained class-incremental experiments, while MMLU-Pro, MATH, and TACO are additionally used for fine-grained long-horizon class-incremental experiments.

The coarse-grained settings group related categories into a small number of broader stages, testing short-horizon class-level adaptation. The fine-grained settings use more stages and therefore test long-horizon stability and cumulative forgetting. Specifically:

- MMLU-Pro is split into 3 coarse-grained groups and 13 fine-grained classes.
- MATH is split into 3 coarse-grained groups and 7 fine-grained classes.

Table 6: Summary of the hierarchical continual MAS evaluation protocol.

Level	Scenario	Datasets or Splits
Task	Cross-task increment	Knowledge QA, Math, Code, Multi-hop QA
Domain	Knowledge QA domain increment	MMLU, MMLU-Pro, AGIEval, GPQA
Domain	Math domain increment	GSM8K, MATH, Olympiad-level Math, TheoremQA
Domain	Code domain increment	APPS, HumanEval, LiveCodeBench, MBPP
Domain	Multi-hop QA domain increment	HotpotQA, 2Wiki, MuSiQue, StrategyQA
Class-Coarse	MMLU-Pro coarse class increment	3 coarse groups
Class-Coarse	MATH coarse class increment	3 coarse groups
Class-Coarse	TACO coarse class increment	3 coarse groups
Class-Coarse	2Wiki class increment	4 reasoning categories
Class-Fine	MMLU-Pro fine class increment	13 fine-grained classes
Class-Fine	MATH fine class increment	7 fine-grained classes
Class-Fine	TACO fine class increment	8 fine-grained classes

- TACO is split into 3 coarse-grained groups and 8 fine-grained classes.
- 2Wiki is split into 4 class-level reasoning categories.

Exact category groups in the protocol builder. The supplementary builder specifies the following ordered coarse groups. Fine streams use the listed categories individually in the same order.

- MMLU-Pro has physics, chemistry, biology, math, and computer science in the first group, health, engineering, business, law, and economics in the second, and psychology, history, and philosophy in the third. These are the 13 selected categories, rather than a claim that the original benchmark has only 13 categories.
- MATH groups Algebra, Prealgebra, and Intermediate Algebra, then Geometry and Precalculus, then Counting & Probability and Number Theory.
- TACO groups Data structures, Range queries, and Amortized analysis, then Dynamic programming and Greedy algorithms, then Sorting, Complete search, and Bit manipulation.
- The four 2Wiki categories are comparison, bridge comparison, inference, and compositional.

TACO matching uses intersection with its skill labels, so an example with multiple labels can belong to several stages. The builder does not establish disjoint class stages merely by assigning stage identifiers. Split isolation and within-stream overlap are separate properties. Stage manifests are needed to quantify the latter.

E.7 Summary of the 12 Continual MAS Scenarios

Table 6 summarizes the 12 continual learning scenarios constructed in our protocol.

E.8 Data Usage and Evaluation

For each continual scenario, the first stage is used as the base task for learning an initial topology generator and constructing the initial shared prior bank. Subsequent stages are treated as continual adaptation tasks. Unless otherwise specified, each new stage provides only a few-shot support set for topology adaptation. After each stage, we evaluate the MAS on the current stage and all previous stages to compute continual learning metrics.

We use task-family-specific evaluation functions to compute normalized MAS utility. For knowledge QA and mathematical reasoning, the utility is based on answer correctness or exact match. For code generation, the utility is based on test-pass correctness. For multi-hop reasoning, the utility is based on exact match or token-level F1, depending on the benchmark. Utilities are normalized to $[0, 1]$ for

the loss formulation. Exact match and token F1 remain different metrics, so cross-family averages summarize the stated evaluation suite and do not make the underlying tasks interchangeable.

The protocol supports the following evaluation goals:

- **Current-stage adaptation:** whether the MAS can construct effective topologies for newly arriving tasks.
- **Historical performance preservation:** whether the MAS maintains performance on previous stages after learning new stages.
- **Long-horizon stability:** whether performance remains stable as the number of continual stages increases.
- **Structural efficiency:** whether the learned topology uses agents and communication edges efficiently.

Source splits and leakage controls. The supplementary builder explicitly maps source splits to adaptation and evaluation files. For MMLU and MMLU-Pro it maps the named source test split to adaptation and validation to evaluation. Datasets without a second source split are partitioned deterministically. Therefore, a file called "train" does not necessarily originate from an official training split. These are continual-protocol results under the constructed splits and should not be presented as untouched official-test results. Within a run, support and evaluation record identities must be disjoint, and preprocessing, hyperparameter selection, and topology retrieval must not use evaluation labels. Exact source revisions, record identifiers, split seeds, and overlap counts are necessary to reproduce that separation.

Asset sources and licenses. The protocol reorganizes the cited public benchmarks. Source splits, repository revisions, access restrictions, licenses, and redistribution permissions are dataset specific and must accompany any released processed files. A public dataset identifier alone does not establish unrestricted redistribution rights. Benchmark descriptions and citation keys are preserved here without asserting a new license audit.

F Baseline Details

This appendix describes the baseline implementations used in our experiments. We compare MASFACT against single-agent reasoning methods, fixed MAS topologies, adaptive MAS topology methods, replay-enhanced continual variants, and plug-and-play MASFACT variants. Comparisons use matched task sequences and evaluation inputs. Comparable MAS variants share the agent pool and execution protocol, while single-agent systems have their own stated inference budgets. Replay adds historical training examples and is reported separately.

F.1 Baseline Categories

We organize the compared methods into four categories.

Single-agent baselines. These baselines do not use a multi-agent communication topology and serve as references for individual reasoning ability. We include:

- **Vanilla**: direct prompting without explicit reasoning traces.
- **CoT**: chain-of-thought prompting, where the model is instructed to generate intermediate reasoning steps before the final answer.
- **Self-Consistency CoT**: multiple chain-of-thought reasoning paths are sampled, and the final answer is selected by majority or consistency-based aggregation.

Fixed MAS topology baselines. These baselines use manually specified communication structures. They do not learn task-specific topologies and therefore provide references for fixed collaboration patterns:

- **Chain**: agents communicate sequentially.

- **Tree**: agents are organized in a hierarchical reasoning structure.
- **Complete Graph**: every agent can communicate with every other agent.

For all fixed-topology baselines, we use the same agent pool and execution budget as MASFACT. The final answer is produced by the designated aggregator agent or by the same aggregation rule used in the main experiments.

Adaptive MAS topology baselines. These baselines construct or adjust MAS communication structures according to the input task. We include:

- **AFLOW (Zhang et al., 2025a)** optimizes executable workflows and appears in the full baseline tables.
- **GDesigner (Zhang et al., 2024)**: a graph-generation based MAS topology designer.
- **AgentDropout (Wang et al., 2025)**: a dynamic agent and communication elimination method.
- **ARGDesigner (Li et al., 2026)**: an autoregressive MAS topology generation method.
- **AGP (Li et al., 2025)**: a pruning-based MAS topology optimizer.
- **MasRouter (Yue et al., 2025)**: a routing-based MAS coordination method.

These methods are originally designed for task-aware or input-aware MAS construction, rather than continual topology learning. We therefore adapt them to our continual protocol by sequentially updating their topology learner or routing policy on each stage using the current support set.

F.2 Continual Adaptation Protocol for Baselines

All trainable MAS topology baselines follow the same continual adaptation protocol. At stage t, the method receives only the current support set S_t, unless a replay variant is explicitly specified. The topology learner is updated on the current stage and then evaluated on the current task and all previously seen tasks. Fixed-topology baselines and single-agent baselines do not update topology parameters; they are evaluated under the same task sequence to provide non-adaptive references.

For adaptive MAS baselines, the sequential update follows each method's original training objective whenever possible. When the original method is not designed for continual learning, we keep its task-level topology learning objective unchanged and apply it sequentially across stages. These sequential adaptations evaluate the chosen continual versions of the baselines and should not be interpreted as their original single-task protocols.

F.3 Replay-enhanced Baselines

For replay-enhanced variants, we maintain a balanced replay buffer. After each stage, the buffer stores the same number of examples from that stage. During training on a new stage, the baseline is trained on the current support set together with replay examples from previous stages. The replay buffer is stage-balanced and does not contain test examples. The main replay series and the separate equal-byte experiment use different budget definitions and must not be conflated. The equal-byte series caps serialized replay storage at the 0.36 MB occupied by the largest reported atom bank.

This replay setting is used only for baseline comparison. MASFACT does not rely on raw historical task replay for topology adaptation; it uses historical topology priors and posterior transfer instead.

F.4 Plug-and-play MASFACT Variants

To evaluate whether MASFACT is tied to a specific topology generator, we attach it to three representative topology learning frameworks:

- **GDesigner + MASFACT**: MASFACT provides aligned historical topology priors and posterior regularization for graph-generation based topology learning.
- **ARGDesigner + MASFACT**: MASFACT provides structural priors that bias the autoregressive topology generation process.

Table 7: Summary of baseline categories.

Method	Category	Learns Topology	Continual Variant
Vanilla	Single-agent	No	No
CoT	Single-agent	No	No
SC-CoT	Single-agent	No	No
Chain	Fixed MAS	No	No
Tree	Fixed MAS	No	No
Complete Graph	Fixed MAS	No	No
AFLOW	Workflow search	Yes	Replay
AgentDropout	Adaptive MAS	Yes	Replay
ARGDesigner	Workflow search	Yes	Replay
GDesigner	Adaptive MAS	Yes	Replay / MASFACT
AGP	Adaptive MAS	Yes	Replay / MASFACT
MasRouter	Adaptive MAS	Yes	Replay

- **AGP + MASFACT**: MASFACT provides an aligned prior over edges and agents, guiding pruning decisions under continual adaptation.

These variants share the same prior-bank construction, FGW alignment, and posterior-transfer regularization. The underlying topology learner remains unchanged except for the additional prior-guided adaptation mechanism. This setup evaluates whether MASFACT acts as a general continual topology learning principle rather than a method specialized to one topology generator. Implementation details are provided in Appendix G.2.

F.5 Prompting and Execution Protocol

All MAS baselines use the same agent role pool and task-specific prompt templates unless otherwise specified. Single-agent baselines use direct task prompts or reasoning prompts. Fixed-topology baselines use the same agents but differ only in their communication structures. Adaptive MAS baselines are allowed to construct or modify their communication structures according to their original mechanisms.

For multi-agent methods, the final answer is produced by a designated aggregator agent or by the common aggregation rule used across experiments. We use the same maximum number of dialogue rounds and the same inference budget for all comparable MAS baselines. Self-Consistency CoT also requires an explicit number of sampled reasoning paths for exact budget matching. This count is not identified in the aggregate result tables.

F.6 Summary of Baselines

Table 7 summarizes the baseline taxonomy.

G Additional Experiments and Analysis

This section retains complete results and the additional analyses. Values are reported on their stated settings. Results from different streams or aggregation protocols are not pooled merely because they share the same metric names.

G.1 Hyperparameter Sensitivity

The recorded sensitivity series varies the FGW fusion weight ρ and posterior penalties $\lambda_{\mathrm{KL}}, \lambda_R$. Its default result is AA 49.1 and AF 2.8, so it is a separate result series from the MATH ablation whose full model has AA 59.10 and AF -0.41.

Sensitivity to FGW fusion weight ρ. At $\rho = 0$, matching uses attributes without relational cost, while $\rho = 1$ uses relations without attribute cost. Table 8 favors an intermediate balance in the

recorded setting. It supports using both sources of information but does not establish insensitivity to ρ across all streams.

Table 8: Recorded FGW sensitivity series. The default configuration is $\rho = 0.50$.

ρ	0.00	0.25	0.50	0.75	1.00
AA	41.8	46.6	49.1	45.4	34.9
AF	8.7	4.9	2.8	3.6	7.2

Sensitivity to conservative adaptation regularizers. Table 9 varies one weight at a time. Smaller penalties increase forgetting in this series, while larger penalties reduce accuracy. This is a setting-specific trade-off, not proof that tuning cannot explain a separate mechanistic trend.

Table 9: Recorded sensitivity to posterior penalties.

Setting	AA	AF
$0.1 \times \lambda_{\mathrm{KL}}$	44.1	7.5
$1.0 \times \lambda_{\mathrm{KL}}$	49.1	2.8
$3.0 \times \lambda_{\mathrm{KL}}$	45.2	4.6
$0.1 \times \lambda_R$	41.0	9.8
$1.0 \times \lambda_R$	49.1	2.8
$3.0 \times \lambda_R$	44.7	6.2

G.2 Plug-and-Play Integration with Existing MAS Topology Optimizers

The integration evaluates a common transfer layer with AGP, GDesigner, and ARGDesigner. These are three topology optimizers, not three different LLM backbones. The interface preserves the agent definitions and execution pipeline while changing how the proposed topology is transferred and adapted.

G.2.1 Integration Principle

For a topology optimizer f_ϕ, the documented interface permits

$$\boldsymbol{B}_t^0 = f_\phi(\mathcal{S}_t), \tag{46}$$

or an edge-score output converted to a weighted scaffold. This support-dependent construction describes the experimental interface and is distinct from the independent-candidate condition of the PAC-Bayes theorem. Historical atoms are retrieved and projected to obtain

$$q_t(\boldsymbol{B}) = \Pi_{\mathcal{B}_t}^{\mathrm{stoch}}(\boldsymbol{B}; \boldsymbol{C}_t + \boldsymbol{R}_t). \tag{47}$$

The selected graph is then passed to the original MAS execution pipeline. The scaffold guides correspondence and does not supply edges copied directly into execution.

G.2.2 Common Interface

The interface receives current support, agent attributes, and an allowed-edge mask. The supplementary adapter converts logits to probabilities with a sigmoid before using them as a weighted scaffold, retains valid probability matrices, and converts discrete graphs to adjacency matrices. Logits and probabilities therefore require different preprocessing. Directly comparing uncalibrated logits from one optimizer with consensus probabilities from another would change the FGW cost scale.

Prior storage, attribute and relation alignment, residual adaptation, and utility-screened atom updates use the same conceptual interfaces across optimizers. Comparisons must still report optimizer-specific construction and adaptation costs. Numerical graph operations do not themselves invoke an LLM, but evaluating additional candidate topologies can require additional calls.

G.2.3 Integration with Pruning-Based Topology Optimizers

For AGP (Li et al., 2025), the pruned graph supplies the scaffold. The atom is then aligned to the current agent space and edited through the residual posterior. The final mask and execution protocol must preserve the baseline's admissible agents and communication rules. Reuse of a historical pattern does not imply that pruning decisions are frozen.

G.2.4 Integration with Routing-Based Topology Optimizers

A routing matrix can be represented as a weighted scaffold after its score convention is specified. ARGDesigner (Li et al., 2026), used in the integration experiments, is an autoregressive graph constructor rather than a generic independent-edge router. Its generated graph or edge scores enter the common scaffold interface. An autoregressive graph distribution is not mathematically identical to the factorized posterior in Appendix D.1. Using the common posterior is a topology-layer adaptation of its output, not a claim that its native probability law is unchanged.

G.2.5 Integration with Graph-Generation-Based Topology Optimizers

For GDesigner (Zhang et al., 2024), the generated graph or edge-score matrix supplies the scaffold. FGW maps a historical consensus to the current agents, and the residual posterior adjusts the resulting reference using support feedback. The LLM agents, prompts, and downstream execution remain matched. The improvement is evaluated empirically in Table 4, without assuming that the transfer layer preserves every property of the original generator.

G.3 Training and Evaluation Protocol

The default few-shot setting uses 200 base examples and 10 support examples per later stage. The backbone is Llama-3.1-8B-Instruct. Within a task family, the agent pool is fixed to isolate topology adaptation. After adapting on support, the utility gate determines whether posterior evidence enters memory. Evaluation follows Appendix D.7, with no gradient update or memory write.

Reported plug-in gains concern matched evaluation settings. They do not by themselves separate the effects of improved initialization, alignment, regularization, and altered search behavior. The ablations and reuse comparisons provide complementary evidence. The fixed-prior theorem remains conditional on how each scaffold and candidate prior is constructed.

G.4 Complete experimental results

The full class, domain, task, and ablation tables below preserve the numerical entries and five-seed standard deviations supplied with the manuscript. Wide tables are split by task family for readability. A dash indicates an unreported entry, not zero performance or zero forgetting. The additional analyses that follow preserve the supplied rebuttal values.

Table 10: Class-level continual learning results with standard deviations. † denotes replay-based variants. Results are averaged over five random seeds.

Method	MMLU-PRO				MATH				TACO				2Wiki				Avg.	
	Few-shot		Standard		Few-shot		Standard		Few-shot		Standard		Few-shot		Standard		AA↑	AF↓
	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓		
Vanilla	26.17±0.12	–	–	–	35.79±0.14	–	–	–	55.15±0.10	–	–	–	39.26±0.11	–	–	–	39.09±0.14	–
CoT-ICL	39.60±0.20	4.01±0.11	–	–	36.83±0.23	4.29±0.14	–	–	58.04±0.24	4.38±0.13	–	–	50.58±0.23	2.61±0.11	–	–	46.26±0.21	3.82±0.12
SC(CoT)-ICL	40.52±0.22	4.95±0.10	–	–	37.88±0.23	6.16±0.11	–	–	59.22±0.22	3.03±0.13	–	–	58.36±0.21	2.15±0.10	–	–	49.00±0.24	4.07±0.12
Chain	36.48±0.14	–	–	–	36.92±0.14	–	–	–	58.46±0.13	–	–	–	50.62±0.14	–	–	–	45.62±0.14	–
Tree	36.82±0.14	–	–	–	36.73±0.14	–	–	–	54.55±0.12	–	–	–	52.74±0.16	–	–	–	45.21±0.13	–
Complete	38.64±0.16	–	–	–	37.63±0.15	–	–	–	59.01±0.13	–	–	–	54.68±0.15	–	–	–	47.49±0.16	–
AFLOW	40.12±0.19	3.97±0.11	42.37±0.20	4.32±0.11	42.18±0.21	3.37±0.11	48.85±0.18	3.92±0.11	62.84±0.22	3.68±0.12	64.12±0.20	4.21±0.10	60.48±0.21	2.90±0.09	61.97±0.20	3.45±0.09	52.87±0.19	3.73±0.11
AFLOW†	42.52±0.18	1.21±0.09	42.89±0.20	2.19±0.12	44.39±0.20	2.03±0.10	49.63±0.19	1.75±0.08	64.75±0.16	2.64±0.12	65.48±0.19	2.13±0.11	63.06±0.17	2.16±0.09	64.73±0.19	2.29±0.12	54.68±0.19	2.05±0.12
AgentDropout	41.07±0.20	3.13±0.12	42.48±0.21	3.83±0.12	43.54±0.22	4.18±0.12	49.88±0.21	4.96±0.10	62.72±0.20	3.17±0.11	63.14±0.22	3.53±0.12	60.07±0.20	1.81±0.10	62.84±0.22	1.97±0.11	53.22±0.19	3.32±0.11
AgentDropout†	43.03±0.19	1.72±0.10	45.84±0.20	1.49±0.12	47.05±0.16	1.56±0.09	51.75±0.16	2.42±0.10	63.38±0.19	1.70±0.08	65.91±0.17	1.42±0.10	62.94±0.20	1.53±0.11	63.58±0.19	1.10±0.08	55.44±0.19	1.62±0.09
MasRouter	45.94±0.21	2.27±0.10	46.17±0.19	2.81±0.09	52.42±0.19	3.14±0.12	54.28±0.20	4.46±0.09	67.62±0.19	3.12±0.10	68.75±0.17	3.70±0.08	65.28±0.20	2.84±0.12	67.51±0.18	3.62±0.10	58.50±0.21	3.25±0.12
MasRouter†	47.40±0.15	1.43±0.10	47.53±0.18	0.83±0.10	55.17±0.17	2.76±0.08	58.56±0.16	1.47±0.09	69.76±0.19	2.89±0.11	71.57±0.16	2.25±0.07	69.16±0.16	1.41±0.08	70.17±0.17	2.06±0.08	61.31±0.16	1.89±0.10
ARGDesigner	44.72±0.18	3.25±0.08	45.52±0.18	3.39±0.12	53.84±0.20	3.62±0.10	56.33±0.19	3.86±0.09	67.36±0.17	3.74±0.12	69.41±0.17	4.28±0.11	62.41±0.17	2.94±0.12	63.91±0.17	3.39±0.12	57.94±0.19	3.56±0.08
ARGDesigner†	46.87±0.16	1.27±0.09	48.69±0.18	1.33±0.10	54.21±0.15	1.88±0.10	57.83±0.17	1.59±0.11	68.82±0.15	2.21±0.08	70.24±0.17	1.92±0.09	65.36±0.15	1.65±0.07	67.51±0.15	2.08±0.10	59.94±0.18	1.74±0.11
GDesigner	45.04±0.20	3.28±0.11	46.72±0.18	3.61±0.09	51.92±0.18	3.89±0.12	53.36±0.17	4.95±0.08	66.41±0.17	4.81±0.08	69.42±0.18	5.26±0.09	63.09±0.20	2.74±0.12	66.27±0.18	3.63±0.11	58.03±0.18	4.02±0.10
GDesigner†	46.16±0.15	1.49±0.11	47.87±0.16	1.22±0.09	54.70±0.14	2.36±0.08	56.18±0.17	1.94±0.11	67.10±0.18	2.22±0.09	72.35±0.18	1.06±0.08	67.79±0.18	1.84±0.11	69.84±0.18	2.17±0.09	60.25±0.18	1.79±0.10
MASFACT	**49.18±0.12**	**-0.19±0.08**	**50.72±0.11**	**0.13±0.07**	**56.07±0.11**	**-0.45±0.04**	**61.68±0.12**	**0.21±0.05**	**70.94±0.10**	**0.12±0.06**	**73.80±0.09**	**-0.11±0.05**	**69.92±0.11**	**-0.26±0.05**	**71.74±0.12**	**0.08±0.07**	**62.73±0.12**	**-0.06±0.04**
Rel. Gains	↑3.76%	↓115.70%	↑4.17%	↓84.34%	↑1.63%	↓128.85%	↑5.33%	↓85.71%	↑1.69%	↓92.94%	↑2.00%	↓110.38%	↑1.10%	↓118.44%	↑2.24%	↓92.73%	↑2.32%	↓103.70%

Table 10 show that MASFACT consistently achieves the highest AA and lowest AF across MMLU-Pro, MATH, TACO, and 2Wiki under both few-shot and standard settings. Relative to the strongest

Table 11: Domain- and task-level continual learning results with standard deviations. † denotes replay-based variants. Results are averaged over five random seeds.

Method	Domain-level continual learning								Task-level continual learning	
	Knowledge QA		Math		Code		Multi-hop QA		KQA→Math→Code→MQA	
	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓	AA↑	AF↓
Vanilla	25.57±0.13	–	29.28±0.14	–	43.78±0.14	–	35.85±0.11	–	30.84±0.15	–
CoT-ICL	34.37±0.21	5.08±0.13	30.19±0.24	6.13±0.13	46.74±0.24	3.29±0.14	38.74±0.24	4.76±0.11	34.90±0.20	5.64±0.14
SC(CoT)-ICL	32.54±0.24	7.46±0.11	33.82±0.22	6.24±0.11	50.46±0.21	3.40±0.14	41.51±0.21	3.24±0.14	37.47±0.22	5.63±0.13
Chain	29.11±0.14	–	30.63±0.16	–	45.64±0.17	–	32.03±0.16	–	32.24±0.17	–
Tree	30.40±0.13	–	34.29±0.17	–	47.12±0.15	–	37.81±0.14	–	34.17±0.15	–
Complete	32.24±0.17	–	35.50±0.13	–	41.71±0.14	–	39.62±0.14	–	36.06±0.14	–
AFLOW	28.45±0.19	7.52±0.12	39.46±0.20	6.89±0.14	48.82±0.21	6.17±0.12	40.18±0.22	4.72±0.12	37.76±0.20	6.45±0.10
AFLOW†	30.21±0.17	3.89±0.12	40.29±0.20	4.16±0.09	54.77±0.18	4.22±0.10	43.68±0.19	1.89±0.12	40.39±0.20	4.12±0.10
AgentDropout	33.15±0.19	5.83±0.13	39.59±0.23	5.62±0.12	48.91±0.20	4.03±0.13	42.94±0.23	5.19±0.14	36.72±0.21	8.12±0.11
AgentDropout†	36.05±0.20	4.07±0.10	41.17±0.18	4.92±0.10	53.58±0.19	2.28±0.10	47.17±0.18	1.04±0.12	40.61±0.17	7.48±0.11
GDesigner	45.62±0.18	6.33±0.11	42.93±0.19	5.14±0.12	54.46±0.17	4.89±0.11	44.92±0.18	5.23±0.11	41.81±0.17	6.95±0.08
GDesigner†	47.73±0.17	4.20±0.07	45.52±0.16	3.57±0.09	57.13±0.17	5.04±0.11	47.97±0.16	1.26±0.08	44.21±0.15	5.15±0.07
MasRouter	46.84±0.20	4.64±0.11	43.86±0.17	4.02±0.09	58.03±0.19	4.18±0.11	45.26±0.17	3.74±0.12	45.50±0.16	7.48±0.11
MasRouter†	49.16±0.15	3.01±0.08	46.28±0.17	3.60±0.07	**62.09±0.18**	3.46±0.08	48.79±0.18	0.72±0.10	49.55±0.14	5.02±0.08
ARGDesigner	47.11±0.17	5.36±0.09	38.94±0.20	3.93±0.11	52.47±0.18	4.09±0.12	46.32±0.18	4.27±0.09	42.18±0.19	6.14±0.12
ARGDesigner†	48.70±0.14	3.16±0.10	40.40±0.14	7.59±0.11	58.41±0.15	2.76±0.08	49.32±0.18	1.90±0.07	49.13±0.17	4.83±0.10
MASFACT	**51.59±0.12**	**0.06±0.05**	**47.91±0.09**	**0.57±0.05**	61.14±0.12	**-2.03±0.06**	**52.31±0.09**	**-0.55±0.04**	**51.67±0.10**	**2.07±0.06**
Rel. Gains	↑4.94%	↓98.01%	↑3.52%	↓84.03%	↓1.53%	↓189.04%	↑6.06%	↓176.39%	↑4.28%	↓49.76%

Table 12: Ablation study on the core designs under MATH 10-shot class-incremental setting. Results include standard deviations over five random seeds.

Method	AA↑	AF↓	ΔAA	ΔAF
MASFACT	**59.10±0.42**	**-0.41±0.18**	–	–
w/o Historical Prior	54.63±0.61	2.36±0.27	-4.47	+2.77
w/o FGW Alignment	55.13±0.47	0.76±0.22	-3.97	+1.17
Euclidean Alignment	54.46±0.55	0.94±0.25	-4.64	+1.35
GW-only Alignment	56.21±0.91	0.38±0.36	-2.89	+0.79
w/o Residual Edit	55.01±0.39	0.03±0.26	-4.09	+0.44
w/o prior KL	54.71±0.29	1.27±0.14	-4.39	+1.68
L2 regularization	56.90±0.46	0.61±0.21	-2.20	+1.02

non-MASFACT AA, MASFACT improves MMLU-Pro by 1.78 and 2.03 points, MATH by 0.90 and 3.12 points, TACO by 1.18 and 1.45 points, and 2Wiki by 0.76 and 1.57 points in the few-shot and standard settings, respectively. Meanwhile, its AF remains close to or below zero across most settings, reaching -0.19 on few-shot MMLU-Pro and becoming negative on standard TACO and few-shot 2Wiki. These results indicate that the accuracy improvements are consistently accompanied by stronger retention, with negative AF further suggesting average backward transfer under the reported metric.

Table 11 shows that MASFACT consistently combines strong adaptation with improved retention across both within-family domain streams and the broader cross-family transition. On Knowledge QA, Math, and Multi-hop QA, it achieves the highest AA and lowest AF, outperforming the strongest non-MASFACT AA by 2.43, 1.63, and 2.99 points, respectively, while reducing AF to 0.06, 0.57, and -0.55. On Code, MASFACT attains the second-highest AA, only 0.95 points below MasRouter†, but achieves the best AF of -2.03. Under the broadest cross-family transition, MASFACT reaches 51.67 AA, exceeding the strongest non-MASFACT baseline by 2.12 points, while reducing AF to 2.07, 2.05 points below the strongest competing retention result. Overall, MASFACT transfers effectively across changing datasets and reasoning families, largely suppressing forgetting in within-family streams and substantially mitigating it under the largest reasoning-paradigm shift.

Table 12 shows that removing any core component reduces AA and increases AF. Removing the historical prior produces the largest AF increase, removing or simplifying FGW weakens cross-space transfer, and replacing prior KL with generic L2 regularization retains substantially more

forgetting. The residual-edit ablation has low AF but loses 4.09 AA points, exposing the expected stability–plasticity trade-off.

Simpler historical reuse baselines. The comparison in Table 13 holds the bank, support budget, agent pool, and execution protocol fixed. Utility-best reuse selects a historically effective graph without task-specific adaptation. Mean topology averages stored adjacency information. Nearest-neighbor reuse omits FGW correspondence. The aligned-prior-only variant uses the projected historical reference without residual editing. Table 13 shows that the aligned prior alone has low

Table 13: Additional historical reuse comparisons. Higher AA and lower AF are preferable.

Method	AA	AF
Utility-best historical reuse	52.84	0.24
Mean historical topology	51.36	0.83
Nearest-neighbor retrieval without FGW	53.19	0.98
FGW-aligned prior only	55.01	0.03
MASFACT	59.10	−0.41

AF but remains 4.09 AA points below the full method. Utility-based, mean-topology, and nearest-neighbor reuse also underperform MASFACT in both AA and AF, supporting the need to combine geometry-aware retrieval with task-specific posterior adaptation. The results do not establish that every possible retrieval or reuse baseline is inferior. The exact nearest-neighbor descriptor and handling of unmatched node cardinalities are not specified in the aggregate comparison.

Scaffold sensitivity. The perturbation study changes the initial scaffold while holding the support set, agent attributes, prior bank, optimizer, and adaptation procedure fixed. Retrieval consistency is the fraction of stages selecting the same top-1 atom as in the clean condition. Table 14 reports the supplied results. Table 14 shows gradual degradation under increasing scaffold corruption. At

Table 14: Scaffold perturbation results. Consistency compares the retrieved atom with the clean condition.

Condition	Consistency (%)	AA	AF
Clean	100.0	59.10	−0.41
Node permutation	98.2	58.94	−0.38
10% edge flips	96.5	58.79	−0.28
20% edge flips	91.7	58.21	−0.12
30% edge flips	84.8	57.46	0.11
Random scaffold	58.3	54.06	0.96

20% edge flips, retrieval consistency remains 91.7% and AA decreases by 0.89 points, whereas a random scaffold lowers consistency to 58.3% and increases AF to 0.96. The scaffold is therefore informative but not the sole transfer signal because attribute matching, utility, dispersion, and residual learning remain active. The study does not isolate the amount of error correction attributable to each subsequent component.

A consistent node relabeling must transform adjacency, attributes, and node measures together. Permuting adjacency alone while holding attributes fixed instead corrupts their correspondence. The reported "node permutation" row does not specify this distinction and is retained as a perturbation result, not as a proof of permutation invariance. The edge-flip denominator, random-graph distribution, and perturbation seeds are also needed for exact reproduction.

Topology-only intervention. At the final stage of the MATH 10-shot class-incremental stream, historical tasks are reevaluated under five execution graphs. LLM weights, agent identities and roles, prompts, questions, aggregation rule, dialogue rounds, inference budget, and evaluation data are held fixed. Only the communication graph is replaced. The stored historical graph was recorded when its task was learned. The drifted graph is reconstructed after the stream. The random graph has the same agent count and similar edge density. The prior-only graph uses projection without

Table 15: Controlled historical-task evaluation on MATH. Accuracy is over old tasks, not the overall AA of the ablation table. Drift values reproduce the supplied normalized measure.

Execution topology	Old-task accuracy	Normalized drift
Stored historical topology	63.10	0.00
Current drifted topology	53.74	0.43
Matched random topology	49.68	0.49
FGW-aligned prior only	61.34	0.07
MASFACT posterior	62.58	0.08

residual adaptation, while the posterior graph uses the final adapted topology. Table 15 shows that replacing the stored historical topology with the drifted topology reduces old-task accuracy by $63.10 - 53.74 = 9.36$ points. The posterior recovers $62.58 - 53.74 = 8.84$ points, or 94.4% of that gap, while the matched random graph performs worst. The prior-only graph recovers most of the gap and residual adaptation improves it further. This controlled replacement demonstrates a causal topology effect under the stated protocol, but does not establish topology as the only source of degradation in other deployments. The drift normalization and repeated-intervention uncertainty are not specified in the supplied summary.

A topology-evolution case study. A separate MMLU-Pro Economics trace illustrates how topology affects access to unchanged agent capabilities. The question requires public cash plus bank reserves. With total money 200, public cash fraction 0.15, and reserve fraction 0.12 on deposits, the answer is

$$0.15(200) + 0.12(200 - 30) = 30 + 20.4 = 50.4. \tag{48}$$

The five registered agents are AnalyzeAgent (A_0), CodeWriting (A_1), MathSolver (A_2), `AdverarialAgent` (A_3), and FinalRefer (A_4). The historical graph is

$$A_0 \to \{A_1, A_2\}, \qquad \{A_1, A_2\} \to A_3, \qquad A_3 \to A_4. \tag{49}$$

It routes parallel numerical and symbolic solutions through a cross-check before the final answer. The drifted graph is

$$A_0 \to A_2 \to A_4, \qquad A_1 \to A_4, \qquad A_3 \to A_4. \tag{50}$$

The two solution paths now bypass A_3. In the supplied trace, A_1 uses 200 rather than deposits 170 to obtain 24, A_2 reports public cash 30, and A_4 outputs 30. Restoring the historical graph allows A_3 to identify reserves $0.12 \times 170 = 20.4$, yielding 50.4.

MASFACT retrieves the parallel-solving and cross-checking pattern, aligns its functional roles, and projects its relations. The reported residual behavior lowers the direct solver-to-final shortcuts while preserving the verification route. Numerical edge probabilities and the full final posterior adjacency are not supplied, so no such values are inferred here. This trace illustrates a possible failure mechanism and recovery, rather than a frequency estimate or proof that KL alone caused the recovery.

Task-order sensitivity. Table 16 reports the original order, reverse order, and five random orders for the 13-stage MMLU-Pro stream. The random-order spread describes order variation in this experiment and should not be substituted for the five-seed standard deviations of the main tables. Table 16 shows that MASFACT retains the strongest AA and lowest AF under the original, reversed,

Table 16: Order sensitivity on the 13-stage MMLU-Pro stream. Each entry is AA / AF.

Method	Original	Reverse	Five random orders
GDesigner	41.67 / 2.34	40.91 / 2.79	41.14 ± 0.53 / 2.63 ± 0.31
GDesigner + replay	37.95 / 2.93	37.42 / 3.28	37.66 ± 0.61 / 3.11 ± 0.27
MASFACT	46.93 / -0.78	46.42 / -0.44	46.55 ± 0.29 / -0.51 ± 0.20

and random curricula. Its random-order standard deviations are also smaller than those of both GDesigner variants in this experiment. The method remains path dependent, and five sampled orders do not characterize all possible curricula.

Bank capacity and retrieval cost. The reported MMLU-Pro bank contains 13 atoms after 200 base examples and 19 after the 12 later stages. Its serialized size is 0.36 MB, including consensus structures, prototype attributes, node measures, historical utility, and dispersion. Table 17 varies capacity while maintaining the same evaluated stream. Table 17 shows that increasing capacity from

Table 17: Capacity and retrieval latency on the MMLU-Pro stream.

Capacity	Final atoms	AA	AF	Retrieval (ms/task)
8	8	45.62	0.04	12.4
13	13	46.28	−0.39	19.8
16	16	46.68	−0.63	24.2
19	19	46.92	−0.77	28.6
Unbounded	19	46.93	−0.78	28.7

8 to 16 improves AA by 1.06 points and changes AF from 0.04 to -0.63, while capacity 16 remains within 0.25 AA points of the unbounded result. Retrieval latency grows from 12.4 to 28.7 ms as the final bank expands from 8 to 19 atoms. The observed saturation supports selective reuse in this stream, not a bound on future growth. Each evaluated graph has at most six agents, and the measured latency should not be extrapolated to much larger graphs without further analysis.

Equal-memory replay and computational cost. The equal-byte comparison limits replay storage to the same reported 0.36 MB as the atom bank. It uses the same optimizer, 10-shot support budget, agent pool of at most six agents, and execution budget. This is a separate replay series from Table 16, whose replay AA is 37.95 rather than 43.15. Table 18 shows that MASFACT adds 7.4% stage time

Table 18: Equal-memory comparison on MMLU-Pro. Time, tokens, and calls are per continual stage. GPU memory is the reported peak.

Quantity	GDesigner	Equal-byte replay	GDesigner + MASFACT
AA	41.67	43.15	46.93
AF	2.34	1.48	−0.78
Time (s/stage)	104.2	168.7	111.9
Peak GPU memory (GB)	9.74	10.47	9.92
Persistent task memory (MB)	0	0.36	0.36
Tokens (thousands/stage)	184.6	287.4	184.6
LLM calls/stage	240	372	240

and 0.18 GB peak GPU memory over GDesigner while leaving tokens and LLM calls unchanged in this measured configuration. Equal-byte replay increases tokens by 55.7%, calls by 55.0%, and stage time by 61.9%. At matched 0.36 MB persistent storage, MASFACT improves AA by 3.78 points over replay and reduces AF from 1.48 to -0.78.

The zero storage entry excludes common model parameters and describes additional persistent task memory. The comparison does not imply that all possible topology searches or integrations incur zero additional LLM calls. Runtime and storage depend on candidate evaluations, retained artifacts, graph size, and implementation. FGW and KL arithmetic alone do not require an LLM invocation.

Plasticity under semantic and structural shifts. Transitions are grouped by the dominant attribute and relational contributions to FGW mismatch. Table 19 compares accuracy on new and old tasks. The groups are a stratified analysis of transitions, not independent random interventions on every possible source of task shift. Table 19 shows that prior-only transfer preserves old-task accuracy but limits new-task plasticity, whereas removing prior KL improves new-task fit at a substantial retention cost. The full method obtains the best new-task accuracy while remaining close to the prior-only old-task accuracy in all three regimes. The atom-creation rate rises from 18% to 74% as shifts become less compatible, which is consistent with bank expansion accommodating new patterns but does not independently prove a unique mechanism. Group thresholds, group sizes, and the creation-rate denominator are required for exact reproduction.

Table 19: New-task / old-task accuracy by shift regime. The final column is the reported atom-creation rate for MASFACT.

Shift	Prior only	Without prior KL	MASFACT	New atoms
Semantic dominant	56.8 / 61.5	61.6 / 55.2	62.5 / 61.0	18%
Structural dominant	52.9 / 61.0	61.0 / 53.6	61.8 / 60.4	47%
Joint distant	47.6 / 60.2	58.4 / 50.8	60.1 / 58.7	74%

Transfer complexity and forgetting. The supplied regime summary reports AF -0.62, 1.18, and 3.74 for close, moderate, and distant transfers, respectively. These regimes use alignment, dispersion, and posterior deviation. Their association with AF supports compatibility as a useful diagnostic. Since posterior deviation is measured after adaptation, the trend is not by itself a prospective prediction or a causal effect of transfer complexity. Hyperparameter sensitivity does not rule out confounding by task difficulty or shift magnitude.

Statistical uncertainty across scenarios. The reported significance analysis first averages each method over five seeds within each of the 12 scenarios. It then compares MASFACT with the reported strongest non-MASFACT baseline in each scenario, using a two-sided exact Wilcoxon signed-rank test and bootstrap intervals with scenarios as resampling units. Table 20 reports favorable differences

Table 20: Reported paired scenario-level improvements. Positive AA differences and negative AF differences favor MASFACT.

Metric	Mean difference	95% bootstrap interval	Exact p	Favorable scenarios
AA	$+2.76$	$[2.18, 3.41]$	4.88×10^{-4}	12/12
AF	-2.31	$[-2.89, -1.76]$	4.88×10^{-4}	12/12

in all 12 scenarios, with confidence intervals excluding zero for both AA and AF. Aggregating seeds first avoids treating 60 runs as 60 independent scenarios, and the reported p value is consistent with $2/2^{12}$ for 12 nonzero paired differences with the same sign under the exact signed-rank null. Reproducing the confidence intervals still requires the scenario-level pairs, comparator identities, bootstrap settings, and seed aggregation records. Because scenarios share datasets and task families, the analysis summarizes this suite rather than population-wide generalization.

H EXPERIMENTAL REPRODUCIBILITY DETAILS

The following separates settings documented in the manuscript and rebuttal from defaults visible in the supplementary implementation. Defaults alone do not identify the configuration that produced a reported table.

Evaluation protocol. LLM agent weights stay fixed throughout the topology-learning experiments. The backbone is Llama-3.1-8B-Instruct. Within each task family, agent pools are fixed, while cross-family tasks may have different agent attributes or pools. Old-task evaluation follows Appendix D.7. Test questions or their answers are not training support, utility-gate evidence, or hyperparameter-selection feedback.

Few-shot and continual settings. The reported few-shot setup uses 200 base examples and 10 support examples for each later stage. The standard columns refer to the corresponding larger adaptation subsets and are not another name for the 10-shot setting.

Metrics. Let $a_{t,i}$ be the evaluation score on task i after stage t, using tasks indexed from 0. Average accuracy is

$$\mathrm{AA}_t = \frac{1}{t+1} \sum_{i=0}^{t} a_{t,i}. \tag{51}$$

To make the manuscript's "best historical performance" convention precise, signed forgetting relative to the best *previous* stage is

$$\mathrm{AF}_t^{\mathrm{best}} = \frac{1}{t}\sum_{i=0}^{t-1}\left[\max_{k=i,\ldots,t-1} a_{k,i} - a_{t,i}\right], \qquad t \geq 1. \tag{52}$$

Excluding the current stage from the maximum permits negative values. A different convention compares with performance immediately after learning each task,

$$\mathrm{AF}_t^{\mathrm{initial}} = \frac{1}{t}\sum_{i=0}^{t-1}(a_{i,i} - a_{t,i}) = -\mathrm{BWT}_t. \tag{53}$$

These two references must not be interchanged. Negative $\mathrm{AF}^{\mathrm{best}}$ implies positive average backward transfer, but the numerical equality to negative BWT applies only to $\mathrm{AF}^{\mathrm{initial}}$. Including the current stage in the maximum would make forgetting nonnegative and contradict the negative values reported here. Scores and differences are in percentage points unless explicitly reported as relative percentages. Drift values are distinct topology measurements.

Random seeds and statistical reporting. The complete result tables state five random seeds and report means with standard deviations. Additional rebuttal tables provide point summaries unless an uncertainty is explicitly shown. Missing deviations have not been inferred from the main tables. Five random task orders are distinct from five stochastic training seeds. The scenario-level test uses seed means rather than treating every seed as an independent task scenario.

Hyperparameters. Table 21 lists identifiable supplementary-code defaults. It is an implementation reference, not a claim that every reported run used these values.

Table 21: Defaults in the supplied core implementation.

Quantity	Default
FGW fusion weight ρ	0.5
Entropy weight ε	0.05
Outer / Sinkhorn iterations	20 / 80
Dispersion / utility retrieval weights	0.1 / 0.1
Average-edge KL / residual weights	1.0 / 0.1
Sampling temperature / threshold	1.0 / 0.5
Atom update rate	0.25
Utility / decision-score thresholds	0.0 / 1.0
Maximum atoms	Unbounded
Numerical confidence parameter	0.05

Table 21 records identifiable implementation defaults but does not establish that every reported experiment used them. In particular, an unbounded atom capacity permits growth, and a zero utility threshold accepts all nonnegative utilities rather than selecting only high-quality evidence. Default coefficients named after stability constants are also not estimates of theoretical constants. Exact reproduction therefore requires the run-specific overrides, loss normalization, KL convention, mask, scaffold inputs, optimizer, atom initialization, and memory-score implementation.

Compute resources. Experiments were conducted on multi-core Sugon x86 CPU nodes with four NVIDIA A100 GPUs allocated per experiment. In this setting, MASFACT used 111.9 seconds per continual stage, 9.92 GB reported peak GPU memory, and 0.36 MB of persistent topology-bank storage, with 184.6 thousand tokens and 240 LLM calls per stage.